ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRÓNICO EN AUTOMATIZACIÓN Y CONTROL

TEMA:

"SISTEMA DE POSICIÓN Y ORIENTACIÓN BASADO EN VISIÓN Y DATOS INERCIALES PARA LA LOCALIZACIÓN DE EFECTIVOS MILITARES EN OPERACIONES ESPECIALES DE COMBATE URBANO"

AUTORES:

AMAGUAÑA QUISHPE, JEFFERSON FABRICIO

TITUAÑA LINCAGO, JONATHAN EDUARDO

DIRECTOR: DR. AGUILAR CASTILLO, WILBERT GEOVANNY

SANGOLQUÍ 2018

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"SISTEMA DE POSICIÓN Y ORIENTACIÓN BASADO EN VISIÓN Y DATOS INERCIALES PARA LA LOCALIZACIÓN DE EFECTIVOS MILITARES EN OPERACIONES ESPECIALES DE COMBATE URBANO"** fue realizado por los señores **AMAGUAÑA QUISHPE, JEFFERSON FABRICIO y TITUAÑA LINCANGO JONATHAN EDUARDO,** el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, Noviembre del 2018

.....

Ing. Wilbert Geovanny Aguilar Catillo PhD.

C. C: 0703844696

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Nosotros, **AMAGUAÑA QUISHPE, JEFFERSON FABRICIO** y **TITUAÑA LINCANGO JONATHAN EDUARDO**, declaramos que el contenido, ideas y criterios del trabajo de titulación: "**SISTEMA DE POSICIÓN Y ORIENTACIÓN BASADO EN VISIÓN Y DATOS INERCIALES PARA LA LOCALIZACIÓN DE EFECTIVOS MILITARES EN OPERACIONES ESPECIALES DE COMBATE URBANO**" ,es de nuestra autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, Noviembre del 2018

.....

Fabricio Amaguaña

C. C: 1721789723

.....

Jonathan Tituaña

C. C: 1721834552

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Nosotros, **AMAGUAÑA QUISHPE, JEFFERSON FABRICIO** y **TITUAÑA LINCANGO JONATHAN EDUARDO**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**SISTEMA DE POSICIÓN Y ORIENTACIÓN BASADO EN VISIÓN Y DATOS INERCIALES PARA LA LOCALIZACIÓN DE EFECTIVOS MILITARES EN OPERACIONES ESPECIALES DE COMBATE URBANO**" ,en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, Noviembre del 2018

.....

Fabricio Amaguaña

C. C: 1721789723

.....

Jonathan Tituaña

C. C: 1721834552

DEDICATORIAS

A mis padres Eduardo y Rebeca por ser el pilar moral y económico que me ha permitido cumplir todas mis metas a lo largo de mi vida académica. A todas las personas que me han brindado sus conocimientos y consejos para poder culminar esta etapa de formación profesional.

Jonathan Eduardo Tituaña Lincango

Dedico el presente trabajo a mi abuela Eusebia (+), mi segunda madre, la persona que me motivo para que no decline en seguir estudiando, la que me impulso y ansiaba verme convertirme en profesional, quien estuvo orgullosa de lo que era y sin duda lo estaría hoy que culmino una etapa más en mi vida profesional.

Y, a mi hijo, Alejandro Amaguaña mi compañero de vida, que llegó en el momento más oportuno convirtiéndose en el motor para impulsar la culminación de mi carrera, quien con su presencia me anima para seguir adelante y trabajar por un mejor porvenir.

Jefferson Fabricio Amaguaña Quishpe

AGRADECIMIENTOS

Agradezco a mis padres Rebeca y Eduardo por su amor y apoyo durante toda mi carrera universitaria. Sus sacrificios diarios han sido mi inspiración para esforzarme por ser una persona de bien. Gracias por permitirme cumplir mis sueños.

A Dios por brindarme salud y sabiduría para llegar hasta este momento tan trascendental en mi vida universitaria.

Agradezco a mis abuelos por enseñarme que el trabajo duro dignifica a las personas. Gracias por enseñarme el don de la humildad y por brindarme su apoyo incondicional en momentos difíciles.

Agradezco a mi tutor de tesis Dr . Wilbert Aguilar por los conocimientos y la ayuda brindada durante el desarrollo de este proyecto de investigación. Durante este tiempo ha sido un referente para seguir con mi preparación académica.

A los amigos y futuros colegas, gracias por hacer de esas duras horas de estudio momentos gratos que llevaré siempre presente.

Jonathan Eduardo Tituaña Lincango

Agradezco a mi madre Mónica Lucia, por ser mi mentora a lo largo de todos estos años de lucha, la persona que nunca perdió la fe en mi y ha sido incondicional, con su amor, su apoyo y sus sabios consejos, gracias por formar y forjar un hombre corrector, respetuoso, honrado y apasionado por sus ideales.

Agradezco a mi padre Tito, de carácter duro, me enseño que el trabajo constante es la base para conseguir lo que uno desea en esta vida, gracias por tu apoyo y tolerancia.

Agradezco a mi hermano por su compañía en los buenos y malos momentos, quien con su admiración y confianza hacia mi persona me llena de motivación a ser un buen ejemplo.

Agradezco a mi abuelita María Dolores, que siempre ha tenido abiertas las puertas de su casa, donde nunca me falto nada, en especial ese cariño y amor en los momentos difíciles.

Agradezco a mis amigos entre ellos, Brayan C. compañero de clase incondicional en cualquier situación que se nos presentó en la universidad y supimos salir adelante. A Cristian A. mi mejor amigo que nunca se descuidó de mi bienestar, por su apoyo y aliento en conseguir este objetivo.

Agradezco a mi tutor de tesis Dr. Wilbert Aguilar, por darme la oportunidad de pertenecer al grupo de investigación de Ejercito, por esa confianza depositada tanto en mi persona como en mis conocimientos, y con el ideal y pensamiento que tiene así sus tesistas, de algún día llamarlo colega, gracias.

Jefferson Fabricio Amaguaña Quishpe

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de investigación presenta un sistema de estimación de posición y orientación basado en algoritmos de visión artificial y datos inerciales tomados de la unidad de medición inercial incorporada en un dispositivo Smartphone. El sistema implementado realiza la estimación de posición y orientación en tiempo real. Para ello se desarrolló una aplicación para sistemas operativos Android que permite capturar las imágenes del entorno y ejecuta los algoritmos de visión artificial. En la implementación del sistema se testean los detectores de puntos característicos en imágenes de Harris, Shi-Tomasi, FAST y SIFT, con el objetivo de encontrar el detector que permita tener un sistema optimizado de modo que pueda ser ejecutado por el procesador de un sistema embebido como son los dispositivos celulares. Para la calcular es desplazamiento de la cámara adherida a un agente móvil se implementó el método de flujo óptico. Adicionalmente se utilizaron datos del giroscopio incorporado en el Smartphone para estimar la orientación del agente. El sistema incorpora una simulación del movimiento estimado dentro de un entorno tridimensional que se ejecuta en un computador. Los datos de posición y orientación se envían desde el celular hacia el computador de forma inalámbrica mediante una conexión Wi-Fi. El entorno tridimensional es una versión digital del bloque central de la Universidad de la Fuerzas Armadas ESPE. Donde se realizaron las pruebas del sistema implementado

PALABRAS CLAVE:

- **HARRIS**
- **SHI-TOMASI**
- **FLUJO ÓPTICO**
- **ANDROID**

ABSTRACT

The present work of investigation presents a system of estimation of position and orientation based on algorithms of artificial vision and inertial data taken from the unit of inertial measurement incorporated in a Smartphone device. The implemented system realizes the estimation of position and orientation in real time. An application was developed for Android operating systems that allows capturing the images of the environment and executes the algorithms of artificial vision. In the implementation of the system the detectors of features points were tested , Harris, Shi-Tomasi, FAST and SIFT, with the objective of finding the detector that allows to have an optimized system so that it can be executed by the processor of a system embedded as are smartphones. To calculate the displacement of the camera adhered to a mobile agent, the optical flow method was implemented. Additionally, gyroscope data incorporated in the Smartphone was used to estimate the orientation of the agent. The system incorporates a simulation of estimated movement within a three-dimensional environment that runs on a computer. The position and orientation data are sent from the smartphone to the computer wirelessly through a Wi-Fi connection. The three-dimensional environment is a digital version of the central block of the Universidad de la Fuerzas Armadas ESPE. Where the tests of the implemented system were carried out.

KEY WORDS:

- **HARRIS**
- **SHI-TOMASI**
- **OPTICAL FLOW**
- **ANDROID**

CAPÍTULO I

1. INTRODUCCION

1.1.Antecedentes

Las operaciones de combate urbano son acciones que realizan las fuerzas militares en un entorno diferente a los escenarios convencionales (montaña, selva, aire, mar), que se ejecutan en ambientes con presencia de bienes, vehículos, personas (W. G. Aguilar, Luna, et al., 2017;Rojas, Moncusi, & Joya, 2013b;W. G. Aguilar, Luna, & Moya, 2018;W. G. Aguilar, Luna, Moya, et al., 2018;W. G. Aguilar, Luna, et al., 2013). Estos ambientes son denominados "urbanos". Los elementos de la distribución urbana deben formar parte del esquema de defensa de la zona de modo que se pueda neutralizar al atacante y mejorar la eficacia de las armas. Sin embargo, en estas operaciones las acciones militares son afectadas por la morfología de las edificaciones urbanas complicando la movilidad del personal militar ("MANUAL DE OPERACIONES DE COMBATE EN AREAS URBANAS by Fernando De Monreal Clavijo - issuu," n.d.). La localización de los efectivos militares es un factor importante, dentro de estas operaciones, que puede verse afectado por las características del entorno.

Esto último se debe a que los sistemas de posicionamiento más comunes basados en satélites (GPS o Galileo) (Hofmann-Wellenhof, Lichtenegger, & Collins, 2001), o en las redes de telefonía móvil ("¿Por qué no funciona el GPS dentro de un edificio? / Pretexsa.com," n.d.),son adecuados para entornos exteriores donde existe visión directa de los satélites o de las estaciones base. Sin embargo, cuando estos sistemas se usan en el interior de edificios, presentan muchos problemas con la atenuación de las señales recibidas y los multitrayectos provocados por los rebotes de la señal.

En los últimos años se han desarrollado varias alternativas para la localización de objetivos en entornos internos o GPS denegados. Una de ellas es la estimación del egomotion basado en imágenes obtenidas por el dispositivo captura desde una perspectiva en primera persona, que permiten el cálculo de la pose (posición y rotación) del dispositivo de captura. Este método de estimación se puede realizar con el uso de cámaras monoculares, cámaras estéreo, cámaras RGBD o sensores LIDAR 3D (Yamaguchi, 2006). En el Reino Unido se desarrolló un sistema de estimación de egomotion mediante el uso de una cámara monocular instalada en un vehículo. En ese sistema, mediante métodos como feature matching y feature tracking, se estimó la velocidad del móvil. Pese a que los valores de velocidad instantánea resultaron ruidosos e imprecisos, los valores de velocidad promedio fueron cercanos a los valores reales demostrando el potencial del sistema para la estimación de egomotion (Bevilacqua, Tsourdos, & Starr, n.d.). Otro sistema sobre estimación de movimiento de un vehículo móvil fue desarrollado en el año 2006. En él se utiliza una cámara monocular y mediante del método conocido como optical flow se logra estimar el movimiento del vehículo (Yamaguchi, 2006).

Un punto favorable para el desarrollo de sistemas y aplicaciones basadas en visión por computador es el avance tecnológico que han tenido los teléfonos inteligentes con altas capacidades de procesamiento en conjunto con múltiples sensores. Un ejemplo de ello es el juego llamado Mozzies (Hannuksela, Barnard, Sangi, & Heikkilä, 2011) desarrollado por Siemens siendo la primera aplicación en usar la cámara de video como sensor para crear un entorno de realidad aumentada.

Paralelamente al desarrollo de los smartphones como dispositivos de procesamiento, el desarrollo de software también ha dado pasos importantes. Desde el año 2000 se ha desarrollado la librería de visión por computador de licencia gratuita. Su principal ventaja radica en que al ser

multiplataforma permite el desarrollo de aplicaciones en distintos softwares de programación como Phyton, C++, Matlab, Unity 3D, entre otros. (“About - OpenCV library,” n.d.)

Tomando en cuenta estos antecedentes, en este proyecto de investigación se propone el desarrollo de una aplicación para dispositivos móviles basada en algoritmos de visión por computador con el fin de realizar la estimación de egomotion para efectivos militares durante operaciones de combate urbano.

1.2. Planteamiento del problema

Desde enero del 2018 la violencia en la frontera norte del país ha aumentado debido a la presencia de grupos subversivos radicales. Los actos delictivos no se llevan a cabo únicamente en la zona selvática como ha sido en la última década, ahora se han trasladado a zona urbanizadas con actos terroristas como el ataque a un comando policial en la zona de San Lorenzo. Este y otros hechos han motivado la participación de efectivos militares en operaciones de apoyo a la Policía Nacional para precautelar la seguridad ciudadana y es en este campo donde las operaciones de combate urbano toman importancia. Estas operaciones de combate tienen como objetivo neutralizar a enemigos hostiles, rescate de rehenes, desactivación de bombas, pero sobre todo, garantizar la paz y tranquilidad ciudadana. Se llevan a cabo en ambientes internos que dificultan el monitoreo de los efectivos militares por parte de los diferentes escalones de mando y control. Se plantea la construcción de un sistema de estimación de *egomotion* de las unidades militares que permita monitorear a las unidades militares durante las operaciones de combate.

Este proyecto busca ser útil para los efectivos militares en operaciones de ambientes urbanos ,en los cuales conocer la ubicación exacta es fundamental ,ya que tomando en cuenta los sucesos

recientemente ocurridos en nuestro país, y la invasión de grupos subversivos radicales, operaciones de rescate o de vigilancia pueden llevarse a cabo con un respaldo de la geolocalización donde se podría monitorear las trayectorias de las unidades y efectivos militares , para ir supervisando su progreso o una eventual extracción de los mismos.

Las aplicaciones pueden extrapolarse hacia sectores como la minería. Debido a su entorno cerrado el uso del GPS queda descartado, y una solución alternativa es la geolocalización por *egomotion*. Esto resultaría útil para situaciones de rescate.

En el ámbito médico puede ser una ayuda significativa, en procedimientos como endoscopias, donde se puede tener una referencia del instrumento con respecto al cuerpo. Permitiendo un análisis más amplio y preciso del estado del paciente.

Este trabajo forma parte del proyecto "Optimización de Recursos y Efectivos Militares a través de una plataforma web que permitan su ubicación monitoreo y visualización" desarrollado por el C.I.C.T.E, cuyo objetivo es desarrollar nuevas tecnologías útiles para mejorar la administración de recursos y para generar un sistema de control y monitoreo de los mismos, de tal manera que las autoridades dispongan de las herramientas mínimas para una adecuada toma de decisiones.

1.3. Alcance del Proyecto

El presente proyecto de investigación se enfoca en la estimación de *egomotion* en entornos internos (Ágila, Link, Smith, & Wehrle, 2012), GPS denegados (Nist & Bergen, 2004), para la localización de efectivos militares durante operaciones de combate urbano utilizando imágenes monoculares e información inercial. El sistema será desarrollado en el entorno de programación Unity3D y ejecutado sobre el sistema operativo para teléfonos móviles, Android. La adquisición

de imágenes se realizará con una cámara monocular (Doctoral, 2015), incorporada en el dispositivo móvil (Smartphone). Adicionalmente se utilizarán datos inerciales proporcionados por la IMU del dispositivo con el fin de mejorar la exactitud en la localización de los efectivos militares.

La ubicación de los elementos militares se visualizará en un entorno virtual(W. Aguilar & Morales, 2016;Amagua, Collaguazo, & Titua, 2018;W. G. Aguilar, Abad, & Ruiz, 2018;Rojas, Moncusi, & Joya, 2013b;W. G. Aguilar, Morales, & Ruiz, 2018), que será una versión digital de un ambiente controlado. Este será ejecutado en un computador, destinado al monitoreo de los efectivos militares.

El entorno donde se desarrollarán las pruebas del sistema desarrollado, serán las instalaciones del bloque B del campus matriz de la Universidad de las Fuerzas Armadas ESPE.

La Figura 1 muestra las etapas del proyecto.

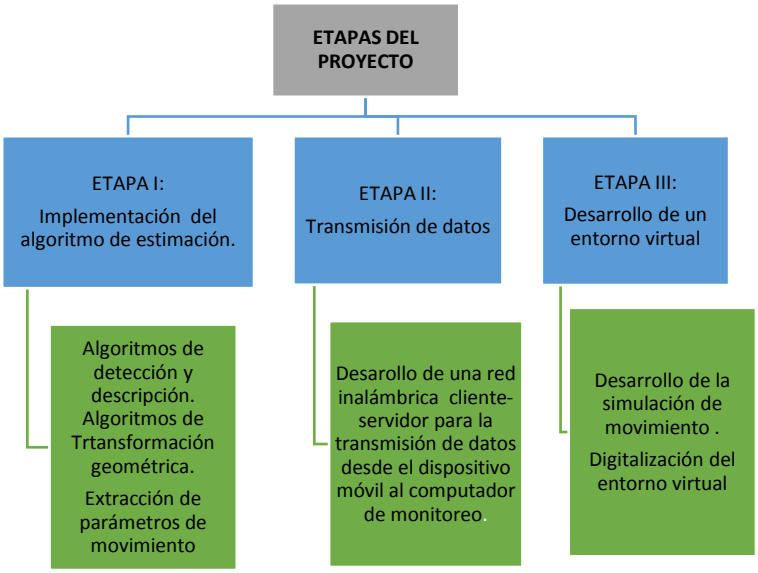

Figura 1. Etapas del trabajo del proyecto.

1.4.Notación

Representación de una imagen	I
Representación de intervalos de tiempos	$k-1$ y k
Transformación geométrica	T
Matrices de rotación	R
Matriz de traslación	t
Posiciones de la cámara	C_n
matriz de calibración	K
Representa un punto en el espacio	X
Vectores de velocidad en eje x	u
Vectores de velocidad en eje y	v
Distancias focales en ejes x y y	(f_u, f_v)
Punto central de la cámara	(c_0, c_0)
Amplitud de la cámara	W
Altura de la cámara	H
Relación de aspecto de la imagen	a
Función de valor máximo	S

Representa el campo de vista de del sensor de la cámara	θ
Coordenadas de un pixel	(x_c, y_c)
Centro Óptico	(c_x, c_y)
Parámetros de distorsión	k_1, k_2
Plano de la imagen captada por la cámara	p_i
Punto donde se mueve un objeto	P_i
Matriz esencial	E
Punto característico de la imagen	p'
Matriz de calibración	K
Matriz de los parámetros extrínsecos e intrínsecos de la cámara	P
Equivalencia es validad hasta a un escalar multiplicativo	$\sim$
Representan la configuración de una cámara	$P_A y\ P_C$
Cantidad de movimiento	$d(\xi, \eta)$
Matriz de deformación de la imagen	D
Traslación del centro de la ventana de seguimiento	d
Disimilitud	ϵ

Ventana de seguimiento de características	W
Componentes de la velocidad	v_x, v_y
Flujo óptico en un pixel	$\check{v}$
Número de iteraciones en RANSAC	N
Número mínimo de puntos de interés	S
Marco del cuerpo	v_b
Rotaciones pequeñas del cuerpo	$\delta\phi, \delta\theta\ y\ \delta\psi$
Límite del ángulo	$\Omega(t)$
Orientación inicial del dispositivo	$C(t)$
Señal de aceleración	$a_b(t)$
Velocidad	$v_g(t)$
Desplazamiento	$s_g(t)$
Cuaterniones (*Quaternions*).	q
Espacio de cuatro dimensiones	R^4
Unidades imaginarias	i, j, k
Vector de en R^3 en giroscopio	v'
Angulo de rotación alrededor del eje X	α

Vector de velocidades angulares medidos por los giroscopios en el eje x,y y z	$\begin{bmatrix} p \\ q \\ r \end{bmatrix}$
Vector de la derivadas de los ángulos *roll,pitch y yaw*	$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$
Tiempo de muestreo	h
Matriz Hessiana	H
Módulo de gradiente	$m\,(x, y)$
Ventana en la posición	$W(x, y)$
Derivadas parciales de *I*	I_x, I_y
Valores propios de la matriz	$\lambda_1 \lambda_2$
Intensidad de un pixel	I_p
Distancia euclidiana	D
Error relativo	ε_R
Error relativo Porcentual	$\varepsilon_R\%$
EMC (Error cuadrático medio)	ECM
Vector de valores estimados	$\widehat{y_i}$
Vector de valores reales	y_i

1.5.Objetivos

1.5.1. General

El objetivo principal de este proyecto es el desarrollar un sistema de ubicación para estimar la posición y orientación de efectivos militares durante operaciones de combate urbano, utilizando algoritmos de *egomotion* e información inercial.

1.5.2. Específicos

- Comparar algoritmos de detección y descripción de puntos de interés.
- Comparar algoritmos de *matching* (búsqueda de correspondencia) o emparejamiento de puntos de interés.
- Implementar los algoritmos de detección, descripción y *matching* que muestren mayor eficiencia.
- Estimar parámetros de movimiento visuales basado en transformaciones geométricas usando los puntos de interés.
- Estimar pose de movimiento basado en la fusión de datos de los parámetros de movimiento visuales y datos inerciales.
- Implementar un sistema de comunicación inalámbrico con arquitectura cliente-servidor.
- Diseñar el entorno virtual para la simulación y el monitoreo de los efectivos militares.

CAPÍTULO II

2. REVISIÓN BIBLIOGRÁFICA Y CONTEXTUALIZACIÓN

Dentro de la robótica en la última década se han desarrollado múltiples aplicaciones de estimación de movimiento con el uso de cámaras de vídeo(W. G. Aguilar & Angulo, 2015;W. Aguilar, Casaliglla, & Pólit, 2017;Lytton, 2014;W. G. Aguilar & Angulo, 2013;W. G. Aguilar, Casaliglla, & Polit, 2017;W. G. Aguilar, Costa-castelló, & Angulo, 2014;Rojas, Moncusi, & Joya, 2013a), este tipo de aplicaciones son conocidas como odometría visual. La odometría visual se ha convertido en una importante alternativa de posicionamiento al sistema convencional del GPS. Existen varios tipos de odometría visual, utilizando cámaras RGB-D(Rojas, Moncusi, & Joya, 2013c;W. G. Aguilar, Rodríguez, & Álvarez, 2013;W. G. Aguilar & Rodr, 2018), estéreo, o un arreglo de cámaras. El último tipo de odometría visual, que se utiliza en este proyecto, emplea una cámara monocular. Sin embargo dentro de la odometría monocular se presentan situaciones durante la estimación del movimiento que se deben específicamente a las condiciones del ambiente como los cambios de iluminación. Estas situaciones crean errores de estimación que se van propagando a lo largo de la trayectoria que realiza el móvil. En los últimos años se ha desarrollado un nuevo enfoque de odometría visual, un método híbrido que utiliza datos inerciales para reducir los errores de estimación y permite qué los resultados sean más exactos. Es por esto que este proyecto y utiliza la odometría visual-inercial para estimar la posición y orientación de efectivos militares durante operaciones de combate urbano.

Este proyecto se divide en tres etapas. La primera etapa comprende la estimación del movimiento utilizando datos de visión e inerciales. En la segunda etapa se desarrolla un sistema de comunicación inalámbrica con el objetivo de transmitir los datos de pose (posición y orientación)

entre el dispositivo móvil y la estación de monitoreo. La tercera etapa consiste en desarrollar un entorno virtual en la estación de monitoreo que permita visualizar la trayectoria de los efectivos. Para la consecución del proyecto se desarrollaron dos algoritmos denominados Cliente y Servidor respectivamente. El algoritmo Cliente se ejecuta sobre un Smartphone y cumple las tareas del sistema de odometría visual- inercial y el envío de datos. El diagrama de la Figura 2 muestra el flujo del sistema de odometría visual implementado. Cabe mencionar que se trata del algoritmo genérico utilizado en este tipo de sistemas.

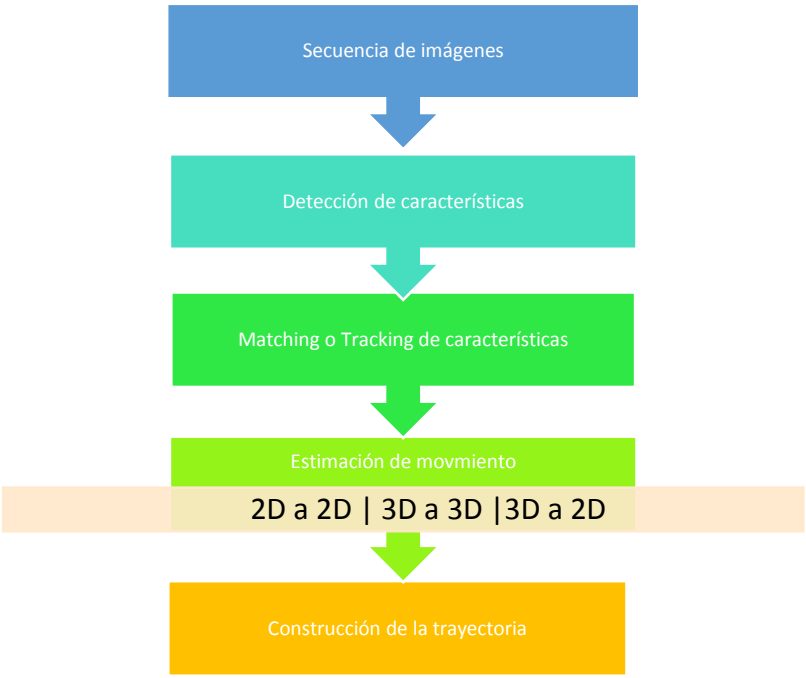

Figura 2. Componentes principales de un sistema de odometría visual.

El algoritmo Servidor se ejecuta en un ordenador y se encarga de recibir los datos de pose enviados por el cliente y realizar la simulación de la trayectoria de los efectivos militares dentro

del entorno virtual. La Figura 1 muestra las etapas del proyecto y el detalle de las tareas realizadas en cada una de ellas. En las siguientes secciones del capítulo se realiza una revisión bibliográfica de la teoría sobre la cual se fundamenta el desarrollo de este proyecto de investigación.

2.1.Odometría Visual

La estimación de movimiento consiste en calcular es desplazamiento en 3D de un objeto, el estudio de estos movimientos se conoce como odometría. Existen varios tipos de odometría que difieren del método utilizado para estimar el movimiento. Por ejemplo para el caso de vehículos con ruedas exclusivamente existe la odometría de rueda que permite la estimación del movimiento a través de los giros de las ruedas del vehículo (B. D. Scaramuzza & Fraundorfer, 2011).

La odometría visual (VO por sus siglas en inglés) por su parte consiste en estimar el movimiento de una agente que puede ser un vehículo, un robot o una persona, mediante el uso de las imágenes tomadas por la cámara o las cámaras que se encuentran adjuntas al agente (B. D. Scaramuzza & Fraundorfer, 2011). Cuando las cámaras se encuentran adjuntas o abordo del agente móvil se conoce como egomotion, dando lugar al término *egomotion estimation*.

El proceso de estimación básicamente consiste en analizar los cambios que se producen entre dos imágenes secuenciales. La odometría visual es un sistema que posee mayor exactitud que métodos como la odometría de rueda (B. D. Scaramuzza & Fraundorfer, 2011), sin embargo existen varias condiciones respecto al entorno a tomar en cuenta durante la estimación del egomotion:

- El entorno debe poseer niveles de iluminación que permitan una eficiente estimación de la trayectoria.

- La escena debe ser estática.
- Debe existir la suficiente textura en el entorno que permita captar los puntos de interés en las imágenes captadas.

Desde 1980 se han desarrollado muchas aplicaciones que trabajan de forma offline es decir que utilizan imágenes grabadas previamente para posteriormente estimar la trayectoria del agente. Sin embargo en la última década desde 2011aplicaciones en tiempo-real han empezado a implementarse (B. D. Scaramuzza & Fraundorfer, 2011). Esto ha dado paso a que la VO se aplique en campos como sistemas autónomos de navegación.

2.1.1. Odometría visual monocular

En la odometría visual existen dos tipos, que difieren del tipo de cámaras utilizado para capturar las imágenes del entorno. La odometría visual estéreo utiliza cámaras estereoscópicas que permiten crear una imagen en tres dimensiones a través de dos imágenes capturadas por las dos lentes que estas cámaras poseen. La odometría visual monocular por otro lado, utiliza una cámara para capturar las imágenes del entorno y estimar el movimiento. Cuando la escena es mucho mayor a la línea base de la VO estéreo (la distancia entre las dos lentes) las imágenes pueden degenerarse, en este caso la VO monocular es usada para estimar la trayectoria (B. D. Scaramuzza & Fraundorfer, 2011). En los últimos años se han investigado sistemas monoculares que han arrojado resultados eficientes en largas distancias (kilómetros) (B. D. Scaramuzza & Fraundorfer, 2011). Otras de las ventajas de la VO monocular es que el movimiento relativo puede ser computado a partir de datos 2-D (puntos).

Sin embargo la VO monocular presenta el inconveniente que para recuperar el movimiento real del agente requiere de una escala relativa que puede ser obtenida mediante otras medidas como los datos arrojados por el acelerómetro de la IMU.

Las aplicaciones desarrolladas mediante este método de odometría se clasifican en 3 categorías,(B. D. Scaramuzza & Fraundorfer, 2011):

- Basadas en características, realiza el seguimiento de características repetibles y destacadas entre cuadros (frames) (B. D. Scaramuzza & Fraundorfer, 2011).
- Basadas en apariencias, utiliza la intensidad de los pixeles y las subregiones aledañas (B. D. Scaramuzza & Fraundorfer, 2011).
- Mixtas, son una combinación de los métodos anteriores (B. D. Scaramuzza & Fraundorfer, 2011).

El primer trabajo de VO monocular en tiempo real fue presentado por D. Nister en (Nist & Bergen, 2004). En él se propone el uso del método de muestra aleatoria de consenso conocido como RANSAC por sus siglas en inglés, para eliminar valores anómalos (outliers) y poder estimar la siguiente pose de la cámara en el siguiente frame consecutivo (Nist & Bergen, 2004). Lo más destacable en su trabajo es la implementación de un solucionador mínimo RANSAC de cinco puntos, lo cual permitía una mejora en procesamiento y exactitud de los algoritmos de estimación visual.

2.1.2. Formulación del modelo matemático del problema de Odometría Visual monocular

Asumimos un agente moviéndose sobre un escenario con una cámara rígidamente adjunta al móvil. La cámara capturará imágenes del entorno cada *k* instantes, las imágenes tomadas se pueden representar por:

$$I_{0:n} = \{I_0, \dots, I_n\} \quad (1)$$

Las posiciones consecutivas de la cámara en los instantes $k-1$ y k respectivamente están relacionadas a la transformación de un cuerpo rígido $T_{k,k-1}$:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \quad (2)$$

Donde $R_{k,k-1}$ y $t_{k,k-1}$ son las matrices de rotación y traslación respectivamente. Las posiciones de la cámara C_n durante el movimiento se pueden determinar mediante:

$$C_n = T_n \, C_{n-1} \quad (3)$$

La Figura 3 muestra gráficamente del problema de odometría visual. En ella se presentan las matrices de la posición de la cámara C_k en cada instante de tiempo, y la matriz $T_{k,k-1}$ representa la matriz de transformación que permite estimar el movimiento de la cámara entre las imágenes I_k, I_{k-1} en los instantes de tiempo $k, k-1$.

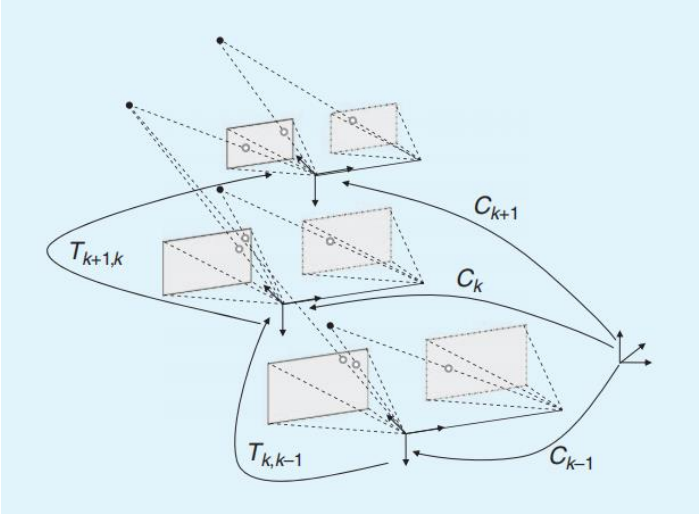

Figura 3. Problema de odometría visual

Fuente: (Visual Odometry) (B. D. Scaramuzza & Fraundorfer, 2011).

Siendo C_0 la pose de la cámara en el instante $k = 0$ que puede ser arbitrariamente impuesta por el usuario. Para recuperar la trayectoria del agente móvil se realiza un incremento pose a pose de los movimientos captados por las poses en los instantes $C_{n-1}\, y\, C_n$. La Tabla 1 muestra las diferentes tipos de transformaciones T_k según los grados de libertad (DoF). La Figura 4 muestra las transformaciones (W. G. Aguilar & Angulo, 2014b;W. G. Aguilar & Angulo, 2014c;W. G. Aguilar & Angulo, 2014a),en un plano 2D.

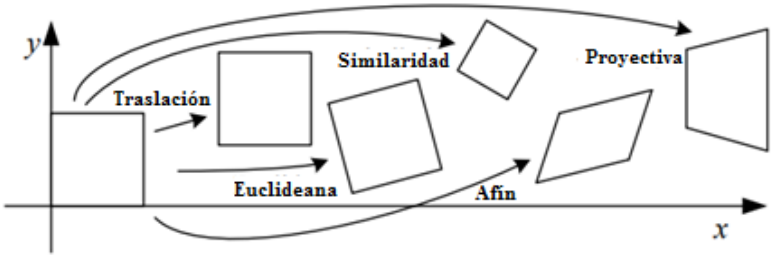

Figura 4. Transformaciones 2D.

Tabla 1.
Transformaciones

Transformación	Matriz	DoF
Traslación	$\begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}$	3
Euclideana	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$	6
Similaridad	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$	7
Affine	$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$	12
Homográfica/Proyectiva	$\begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$	15

Fuente: (Visual Odometry) (B. D. Scaramuzza & Fraundorfer, 2011).

2.1.3. Modelado y calibración de la cámara

2.1.3.1. Modelo de la cámara Perspectiva

En este modelo se asume el sistema de proyección de agujero de alfiler. Describe matemáticamente la correspondencia entre los puntos observados en el mundo real (puntos 3D) y los pixeles captados por la imagen de la cámara (puntos 2D) (Fraundorfer & Scaramuzza, 2012). La Figura 5 muestra el modelo de la cámara perspectiva, el cubo representa la cámara y el agujero representa el lente de la misma conocido como centro de proyección. Allí se puede observar cómo la cámara representa los objetos captados en el mundo real. La imagen se forma con la intersección de los rayos de luz con el plano focal (B. D. Scaramuzza & Fraundorfer, 2011). La correspondencia de los puntos se representa por la ecuación matricial:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KX \qquad (4)$$

Donde X representa un punto en el espacio con coordenadas $X = [x, y, z]^T$ y el punto $p = [u, v]^T$ la proyección del punto sobre el plano focal, cuya medida se da en pixeles,(B. D. Scaramuzza & Fraundorfer, 2011). Y la matriz K (matriz de calibración) contiene los valores intrínsecos de la cámara como las distancias focales (f_u, f_v) respecto a los ejes x y y respectivamente, el punto central de la cámara (c_0, c_0). Obteniendo la siguiente ecuación matricial que permite obtener los puntos 2D de proyección de un objeto en el mundo real.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & s & c_0 \\ 0 & f_v & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (5)$$

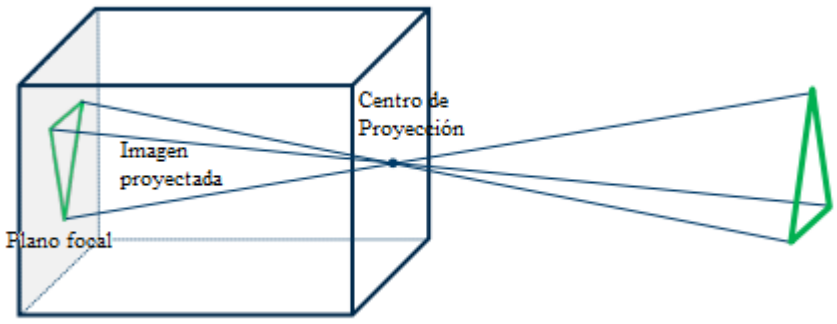

Figura 5. Modelo cámara perspectiva

Fuente: (Opsahl, n.d.).

Los parámetros intrínsecos del modelo describen la transformación de las coordenadas de una imagen normalizada a coordenadas de imagen que generalmente se mide en pixeles. Estos valores

describen aspectos físicos de la relación entre la imagen normalizada y la imagen proyectada sobre el plano focal como muestra la Figura 6.

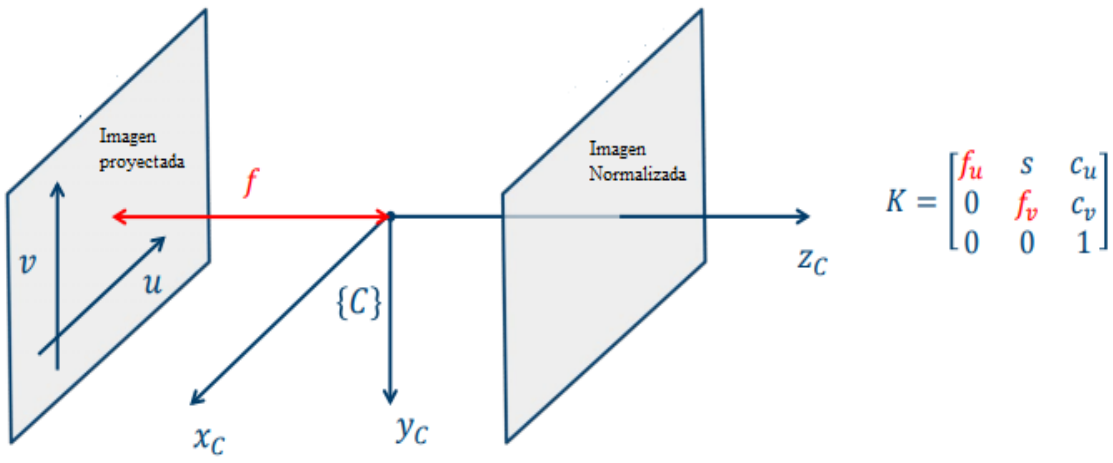

Figura 6. Parámetros intrínsecos.

Fuente: The perpective camera model (Opsahl, n.d.)

Finalmente para el cálculo de la perspectiva de la cámara en un instante k respecto a la perspectiva en un instante $k-1$ se requiere de los parámetros extrínsecos de la cámara, con el objetivo de determinar la correspondencia de los puntos (características de las imágenes) entre dos imágenes consecutivas I_k, I_{k-1} . Estos parámetros extrínsecos son los datos de rotación, traslación y escala que se trataron en la sección anterior, en la matriz de transformación. De esta forma el modelo que caracteriza la perspectiva de la cámara de la siguiente manera.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & s & c_0 \\ 0 & f_v & c_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (6)$$

- **Distancias Focales**

Uno de los principales problemas en algoritmos de odometría visual son las unidades para expresar las distancias focales. Muchas cámaras convencionales expresan sus dimensiones W (amplitud) y H (altura) en milímetros. Sin embargo es conveniente expresar estas dimensiones en píxeles de tal forma que las distancias focales se expresen también en pixeles y se puedan usar en los algoritmos de calibración. En la Figura 7 se muestra gráficamente la relación que existe entre la distancia focal y el ancho del sensor de la cámara.

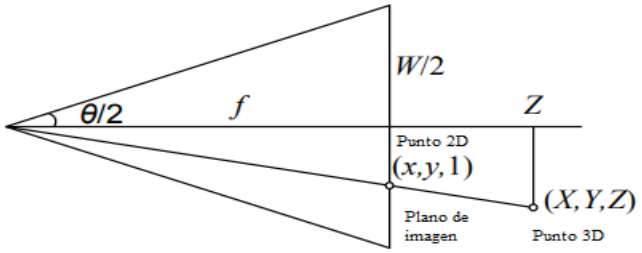

Figura 7. Relación entre el ancho de la imagen y la distancia focal.

Fuente: Computer visión (Szeliski, 2010)

Matemáticamente esta relación queda definida por la siguiente ecuación:

$$\tan\frac{\theta}{2} = \frac{W}{2} \quad \text{ó} \quad f = \frac{W}{2}\left[\tan\frac{\theta}{2}\right]^{-1} \qquad (7)$$

Donde θ representa el campo de vista de del sensor de la cámara.

Existe otra posibilidad para encontrar las distancias focales, para ello se debe escalar los pixeles de modo que varíen en el rango de [-1,1) a lo largo de la dimensión más grande del sensor (W o H) y en el rango de [$-a^{-1}$, a^{-1}) a lo largo de la dimensión más corta, donde $a \geq 1$ es la relación de

aspecto de la imagen (Szeliski, 2010). Para completar se usan las coordenadas del dispositivo normalizado:

$$x'_s = \frac{2x_s - W}{S} \quad , \quad y'_s = \frac{2y_s - H}{S} \quad donde\ S = \max(W, H) \tag{8}$$

Una de las ventajas de esta opción es la independencia que existe entre la resolución de la imagen con las distancias focales y el centro de la imagen. Esta característica es muy útil cuando se trabaja con algoritmos que utilizan pirámides de imágenes para la detección de puntos característicos. Si hacemos $S = W = 2$ se puede obtener una relación adimensional.

$$f^{-1} = \tan\frac{\theta}{2} \tag{9}$$

De esta forma para expresar f , en unidades adimensionales, en pixeles basta con multiplicar por un factor de $W/2$ y para convertir f expresado en pixeles a una distancia equivalente a 35mm, se multiplica por $35/W$. Donde 35mm es la amplitud focal del sensor en las cámaras convencionales (Szeliski, 2010).

2.1.3.2.Calibración de la cámara

Para obtener buenos resultados en la estimación de la pose de la cámara, es necesario mejorar la exactitud de la medida de los parámetros extrínsecos e intrínsecos de la misma para esto la calibración de la cámara es un paso fundamental para la estimación del *egomotion* (B. D. Scaramuzza & Fraundorfer, 2011).

Existen varios métodos de calibración, siendo el más utilizado el método de minimización de mínimos cuadrados. Para esto se utiliza un tablero de ajedrez, el usuario debe tomar varias imágenes del tablero en diferentes posiciones, asegurándose que el tablero entre en todo el marco de la imagen. Este proceso permite estimar adicionalmente la distorsión radial del lente de la cámara a través de la distancia entre las esquinas de los cuadros del tablero como muestra la Figura 8 ("OpenCV: Camera Calibration," n.d.).

Figura 8. Calibración de la cámara.

Fuente: Camera calibration ("OpenCV: Camera Calibration," n.d.)

- **Distorsión radial del lente**

El modelo de agujero de alfiler descrito anteriormente asume que las cámaras siguen un modelo de proyección lineal. Es decir que las líneas rectas captadas por la cámara en el mundo real resultan en líneas rectas en la imagen. Sin embargo en lentes gran angular se presentan distorsiones que se manifiestan como líneas curvas hacia fuera de la imagen o hacia el centro de la misma, estas

distorsiones afectan la exactitud en aplicaciones basadas en visión por. Como se muestra en la **Figura 9**, en ella se muestran 3 diferentes distorsiones radiales que se pueden presentar. En la imagen de la izquierda se produce una distorsión tipo barril, en la imagen central una distorsión tipo almohadilla y la tercera imagen una distorsión tipo ojo de pez.

Figura 9. Distorsión radial del lente.

Fuente: Computer vision (Szeliski, 2010)

En la práctica compensar una distorsión radial no resulta difícil, para ello un modelo cuadrático puede producir buenos resultados (Szeliski, 2010). Si asumimos las coordenadas de un pixel como (x_c, y_c), luego de una división perspectiva pero antes de escalar por la distancia focal y movida por el centro óptico (c_x, c_y), las coordenadas de pixel quedan definidas como:

$$x_c = \frac{r_x \cdot p + t_x}{r_z \cdot p + t_z} \quad (10)$$

$$y_c = \frac{r_y \cdot p + t_y}{r_z \cdot p + t_z} \quad (11)$$

El modelo de distorsión radial asume que estas coordenadas pueden ser desplazadas hacia fuera o hacia el centro de la imagen, por una cantidad proporcional a su distancia radial (Szeliski, 2010). Usando polinomios de bajo orden tenemos:

$$x'_c = x_c(1 + k_1 r_c^2 + k_2 r_c^4) \quad (12)$$

$$y'_c = x_c(1 + k_1 r_c^2 + k_2 r_c^4) \quad (13)$$

Donde $r_c^2 = x_c^2 + y_c^2$ y k_1, k_2 son los parámetros de distorsión. Finalmente las coordenadas del pixel se calculan:

$$x_s = f x'_c + c_x \quad (14)$$

$$y_s = f y'_c + c_x \quad (15)$$

2.2.Estimación de Movimiento

En la sección anterior se definió el modelo de perspectiva de la cámara con el objetivo de mostrar las ecuaciones que servirán para el cálculo de la trayectoria del agente. Si bien la ecuación 5 muestra la matriz de transformación T_k de la cámara entre dos frames o cuadros consecutivos, al tratarse de egomotion la cámara se encuentra adherida al agente y por lo tanto representa el cambio de pose (posición y giro) del mismo.

El cambio de posición de la cámara se calcula mediante la matriz T_k entre dos imágenes consecutivas I_k, I_{k-1} correspondientes a los instantes de tiempo $k, k-1$, respectivamente (B. D. Scaramuzza & Fraundorfer, 2011). La concatenación de estos movimientos permite estimar la trayectoria del agente a través del cómputo de conjuntos características f_k, f_{k-1} correspondientes

a las imágenes I_k, I_{k-1}. Los conjuntos de características son puntos de alto contraste que se encuentran en una imagen.

La matriz de transformación se determina mediante el cálculo de la correspondencia entre los conjuntos de características f_k, f_{k-1}.Existen tres métodos que permiten estimar la correspondencia y dependen del tipo de características (puntos 2D o 3D) que se usan en cada método.

- Correspondencia 2D a 2D, los puntos característicos de las dos imágenes consecutivas se especifican en coordenadas de una imagen 2D generalmente se usan los pixeles como unidad de medida (B. D. Scaramuzza & Fraundorfer, 2011).
- Correspondencia 3D a 3D, en cada instante k es necesario triangular los puntos f_k, f_{k-1} con el uso de cámaras estereoscópicas (B. D. Scaramuzza & Fraundorfer, 2011).
- Correspondencia 3D a 2D, los puntos f_{k-1} se especifican en 3D y los puntos f_k son proyecciones en una imagen 2D. Para el caso de cámaras monoculares los puntos 3D se triangulan mediante el uso de dos imágenes (I_{k-1}, I_{k-2}) captadas por cámaras adyacentes (B. D. Scaramuzza & Fraundorfer, 2011).

2.2.1. Correspondencia de características 2D – 2D

Si asumimos el problema de un objeto rígido moviéndose desde el punto $P_i = (x_i, y_i, z_i)$ hasta el punto $P' = (x_i\,', y_i', z_i,)$ en el espacio. La proyección de cada punto sobre el plano de la imagen captada por la cámara será $p_i = (X_i, Y_i, 1), p'_i = (X_i\,', Y_i\,', 1)$ respectivamente, y la correspondencia que existe entre estos dos puntos viene dada por la ecuación (16).

$$p'_i = Rp_i + t \quad (16)$$

Donde $R\ y\ t$ representan las matrices de rotación y traslación respectivamente. Entonces el problema a solucionar es dado N correspondencias $(p_i, p_i)_{i=1,\dots,N}$ determinar R y t en cada caso (Huang & Netravali, 1994).

La relación geométrica que existe entre dos imágenes consecutivas I_k, I_{k-1} capturadas por una cámara calibrada, es decir con parámetros intrínsecos conocidos, se encuentra descrita por la matriz esencial conocida como matriz E (B. D. Scaramuzza & Fraundorfer, 2011). Esta matriz está formada por las matrices R y t :

$$E_k \simeq \widehat{t_k}\, R_k \quad (17)$$

Donde $t_k = [t_x, t_y, t_z]^T$ y

$$\widehat{t_k} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

La propiedad principal de la correspondencia 2D a 2D es que posee una restricción epipolar que determina la línea en la que el punto característico correspondiente p' de p se encuentra en la otra imagen (B. D. Scaramuzza & Fraundorfer, 2011), como muestra la Figura 10. Esta restricción epipolar se formula de la siguiente forma:

$$p'^T E\, p = 0 \quad (18)$$

Donde p' es un punto característico de la imagen I_k y p es un punto característico de la imagen I_{k-1}. De esta forma la matriz E permite determinar la correspondencia 2D a 2D.

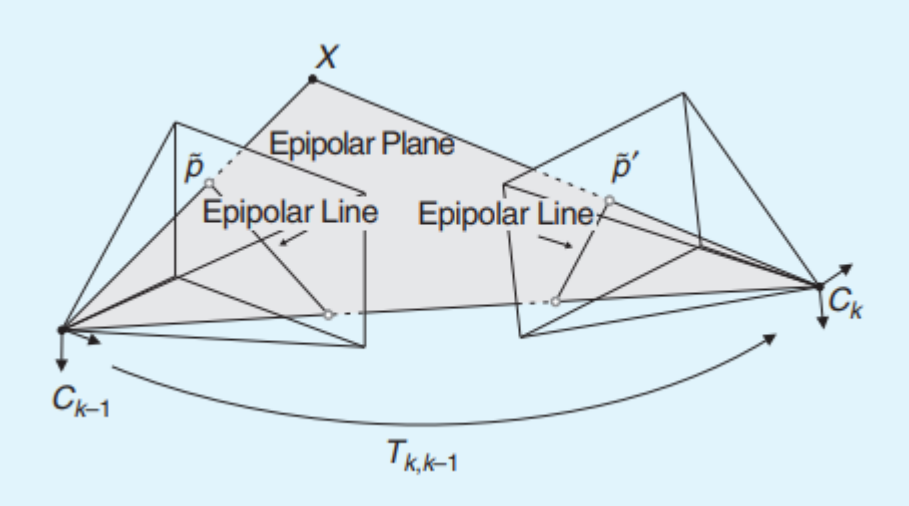

Figura 10. Restricción epipolar.

Fuente: (Visual Odometry) (B. D. Scaramuzza & Fraundorfer, 2011).

2.2.1.1.Cálculo de R y t

Una vista con un centro de proyección finito se describe de la siguiente forma:

$$P = K[R|t] \tag{19}$$

Donde K es la matriz de calibración y P se denomina matriz de cámara debido a que contiene los parámetros extrínsecos e intrínsecos de la misma. Sea D una matriz 3x3 de la siguiente forma:

$$D = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El método para recuperar R y t de la matriz esencial se basa en el siguiente teorema:

Teorema 1: si asumimos la descomposición de la matriz esencial como $E \sim U\, diag(1,1,0) V^T$, donde U y V son escogidas de tal forma que $\det(U) > 0$ y $\det(V) > 0$. Entonces $t \sim t_u \equiv [u_{13}\ u_{23}\ u_{33}]^T$ y R es igual a $R_a = UDV^T$ o $R_a = UD^TV^T$ [19].

Donde $\sim$ significa que la equivalencia es validad hasta a un escalar multiplicativo. Cualquier combinación de R y t puede satisfacer la ecuación 8 (Nist, n.d.). Se asume que para la primera imagen la matriz de cámara resulta de la combinación de $P = [I|0]$ es decir R es una matriz identidad de 3x3 y t es un vector 0 de 3x1 (Nist, n.d.). Las cuatro posibles soluciones serían:

$$P_A = [R_a|t_u] \quad (20)$$

$$P_B = [R_a|-t_u] \quad (21)$$

$$P_C = [R_b|t_u] \quad (22)$$

$$P_D = [R_b|-t_u] \quad (23)$$

Para seleccionar la respuesta correcta se utiliza un punto por triangulación y se selecciona la solución para la cual el punto se encuentra frente a una de las cámaras. $P_A y\, P_C$ Representan la configuración de una cámara $P_B y\, P_D$ representan otra la configuración de otra cámara.

2.3.Detectores y descriptores de puntos de interés (puntos característicos)

La forma de calcular el movimiento es un parte fundamental para cálculo de odometría visual, en ese apartado se calcula el movimiento de las imágenes entre un instante t y una anterior en el instante $t-1$. Uniendo todas las capturas y los diferentes cálculos se puede llegar a conocer la trayectoria realizada por la cámara y por lo tanto de su portador. Para las matrices de transformación entre dos imágenes T_t y T_{t-1}, se debe empezar por definir un conjunto de características (*features points*) f_t y f_{t-1} , estos conjuntos son tomados en los instantes t y $t-1$ de cada imagen respectivamente.(Doctoral, 2015)

Estas características pueden ser expresadas en 2D o 3D, según sea la definición se usa diferentes métodos de extracción y pueden ser expresados por puntos o líneas dentro de la imagen, aunque los más usados son los puntos.

2.3.1. Detección de puntos de interés.

Para la detección de puntos de interés se puede seguir dos metodologías, la primera resulta ser más intuitiva, y se basa en la extracción de puntos de la imagen inicial y luego se realiza la búsqueda de dichos puntos en las imágenes posteriormente capturadas este método se lo emplea para movimientos pequeños(W. G. Aguilar & Angulo Bahón, 2013;W. G. Aguilar & Angulo, 2012;W. G. Aguilar & Angulo, 2012). Por otro lado, se puede encontrar puntos en cada imagen de forma independiente y luego buscar sus correspondencias; este método es usado para movimiento con distancias más largas.(Jorge et al., 2014)

En la detección de puntos lo que se busca es que sean fácilmente distinguibles y comparables, existen dos tipos de puntos, esquinas y regiones de la imagen que pueden ser caracterizados por su color o texturas también llamados *blobs*.

Propiedades que caracterizan a un buen detector:

- Precisión en la posición dentro de la imagen
- Repetibilidad
- Eficiencia en la ejecución
- Robustez (ruido o desenfoque)
- Distinción (características claramente identificables)
- Variación nula (cambios de luminosidad)

- Cambios geométricos (rotación, escala y distorsiones de perspectiva)

Los detectores de esquinas más usados en odometría visual son Moravec, Forsher, Harris, Shi-Tomasi y Fast. Mientras que los detectores de blobs más recurrentes son SIFT, SURF y CENSURE por mencionar algunos.(Fraundorfer & Scaramuzza, 2012) Cual quiera de los dos tipos tiene sus ventajas y desventajas los detectores de esquinas son más rápidos, pero generan incertidumbres en diferenciar las características por lo que surgen errores en el seguimiento y son menos robustos; en contra parte los detectores de blobs detectan con más facilidad las diferencias sin embargo son más lentos y demandan de un coste computacional más alto que los anteriores.(Doctoral, 2015) Dado los lineamientos del desempeño de los detectores no es una elección que se la tome con poco criterio si no va a depender del trabajo y las condiciones en que se vaya a realizar la estimación de movimiento.(Fraundorfer & Scaramuzza, 2012)

Tabla 2
Comparativa de detectores de características

	Esquinas	**Blobs**	**Rotación**	**Escala**	**Repetibilidad**	**Localización**	**Robustez**	**Eficiencia**
Harris	X		X		+++	+++	++	++
Shi-Tomasi	X		X		+++	+++	++	++
FAST	X		X	X	++	++	++	++++
SIFT		X	X	X	+++	++	+++	+
SURF		X	X	X	+++	++	++	++
CENSURE		X	X	X	+++	++	+++	+++

Fuente:(Fraundorfer & Scaramuzza, 2012)

2.3.1.1.Buenas características para Tracking (GoodFeaturesToTrack)

Los sistemas de visión basados en detección y seguimiento (tracking) de características dependen esencialmente de la calidad de las mismas. Entre imágenes consecutivas pueden aparecer puntos que no se encontraban en el frame anterior o incluso desaparecer puntos en la siguiente imagen, debido a cambios de iluminación en el ambiente o la textura de la región sobre la cual se está capturando la imagen. Shi-Tomasi en (Tomasi, n.d.) desarrollan un método que permite verificar la calidad de los puntos característicos entre dos imágenes consecutivas, para ello utilizan una medida de disimilitud. Si la disimilitud entre dos puntos crece mucho, el punto es descartado.

Los movimientos de una cámara pueden ser descritos como:

$$I(x, y, t + \tau) = I(x - \xi(x, y, t, \tau), y - \eta(x, y, \tau)) \quad (24)$$

Una imagen tomada en el instante $t + \tau$ puede ser obtenida, moviendo cada punto de la imagen actual una cantidad de movimiento adecuada (Tomasi, n.d.), este modelo de movimiento se lo conoce como modelo de traslación puro. La cantidad de movimiento $d(\xi, \eta)$ es también denominado desplazamiento de un punto en $X = (x, y)$. Las variaciones de movimiento son incluso detectadas cuando se usa una ventana de seguimiento pequeña. Sin embargo una representación mediante un movimiento afín es una mejor representación que un modelo de traslación puro:

$$\delta = DX + d \ \ (modelo\ afín) \quad (25)$$

Donde D es la matriz de deformación de la imagen:

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$$

d Es la traslación del centro de la ventana de seguimiento y X las coordenadas de imagen medidas respecto del centro de la ventana (Tomasi, n.d.). Entonces si asumimos que un punto X en una imagen I se mueve a un punto $AX + d$ en una segunda imagen J tenemos la correspondencia:

$$J(AX + d) = I(X) \quad (26)$$

Donde $A = 1 + D$, 1 es una matriz identidad de 2x2. Sin embargo durante el tracking de características se asumen tazas de muestreos cortos entre frames para capturar imágenes y por lo tanto la matriz de deformación D se asume como cero:

$$\delta = d$$

Shi-Tomasi en (Tomasi, n.d.) demostraron que la mejor combinación de los dos modelos de movimiento son: modelo afín para comparar características y el modelo de traslación pura para tracking de características. De esta forma el problema de determinar los parámetros de movimiento es encontrar las matrices A y d que minimizan la disimilitud ϵ (Tomasi, n.d.):

$$\epsilon = \iint_W [J(AX + d) - I(X)]^2 w(X) d(X) \quad (27)$$

Donde W es la ventana de seguimiento de características y $w(X)$ es una función de peso.

La librería de visión por computadora integra la función *goodFeaturesToTrack*() basada en el principio de disimilitud desarrollado por Shi-Tomasi para la selección de características. El algoritmo de detección implementado en la función es una modificación del detector de esquinas de Harris. En él se utiliza una versión alterada de la función de puntuación del detector de Harris

que determina si la ventana de seguimiento contiene o no una esquina ("Harris Corner Detection — OpenCV 3.0.0-dev documentation," n.d.).

$$R = \min(\lambda_1, \lambda_2) \quad (28)$$

Donde λ_1, λ_2 son los valores propios de la matriz formada por la función de pesos y las imágenes derivadas en las direcciones de x y y. En la Figura 11 se muestra gráficamente la región (azul) de valores para los cuales una característica es considerada una esquina. En otras palabras cuando $\lambda_1 \ \lambda_2 > \lambda_{min}$. Donde λ_{min} es el umbral de calidad de las características entre [0,1].

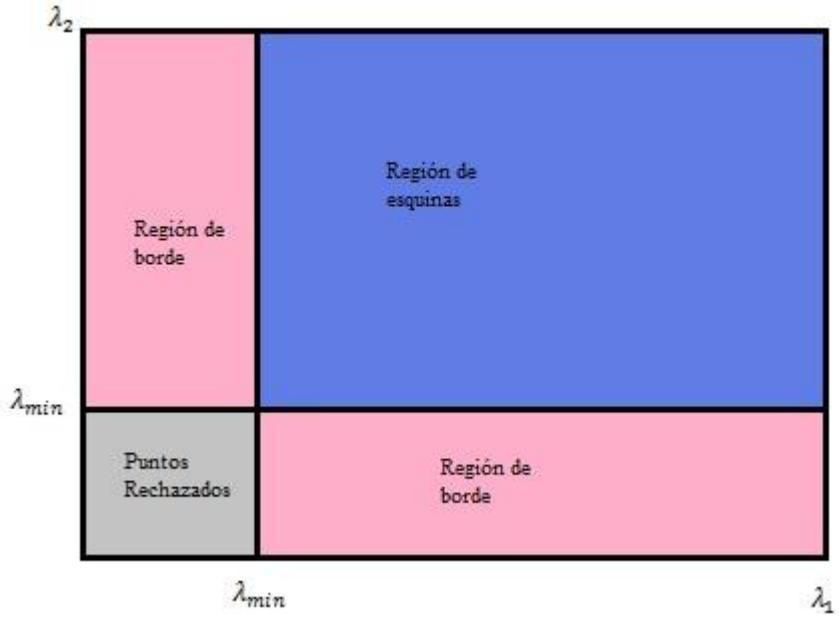

Figura 11. Región de selección de esquinas

Fuente: Harris detector ("Harris Corner Detection — OpenCV 3.0.0-dev documentation," n.d.)

2.3.2. Descriptores de puntos de interés.

Realizada la detección de los puntos de interés el paso a seguir es identificar dichas características en la imagen, esto se logra mediante los descriptores. Para ello la conformación de un descriptor es mediante la región que esta alrededor del punto de interés, así cuando se busquen

similitudes entre puntos se compara el píxel del punto característico y los pixeles vecinos. (Jorge et al., 2014)

El descriptor óptimo será aquel que tome como valores a identificar los valores de intensidad del punto de interés y de los pixeles vecinos. En base a esto la comparación de los descriptores se los hace con funciones de error tal como el sumatorio de diferencias de cuadrados de intensidad (SSD) o la correlación cruzada normalizada (NCC). Una alternativa de similitud es la trasformada Censure que convierte las áreas de la imagen en u vector binario donde se alojan la intensidad de los pixeles vecinos tanto superior o inferior al punto central, la similitud entre áreas se obtiene mediante la distancia de Haming.(Fraundorfer & Scaramuzza, 2012)

Sin embargo, estos descriptores son muy limitados debido a sus variaciones en orientación, escala y punto de vista por lo que solo son usados cuando la visión está próximo a los objetos.

El descriptor SIFT consiste en formar un histograma de gradientes de orientaciones locales, por lo que el área alrededor del punto se lo divide en 4x4 celdas, para cada cuadrante se forma un histograma de 8 gradientes de orientación. SIFT ha demostrado ser más estable ante variaciones como iluminación, rotación, escala o cambios de visión, su desempeño es mejor cuando se usa en blobs que en esquinas.(Doctoral, 2015)

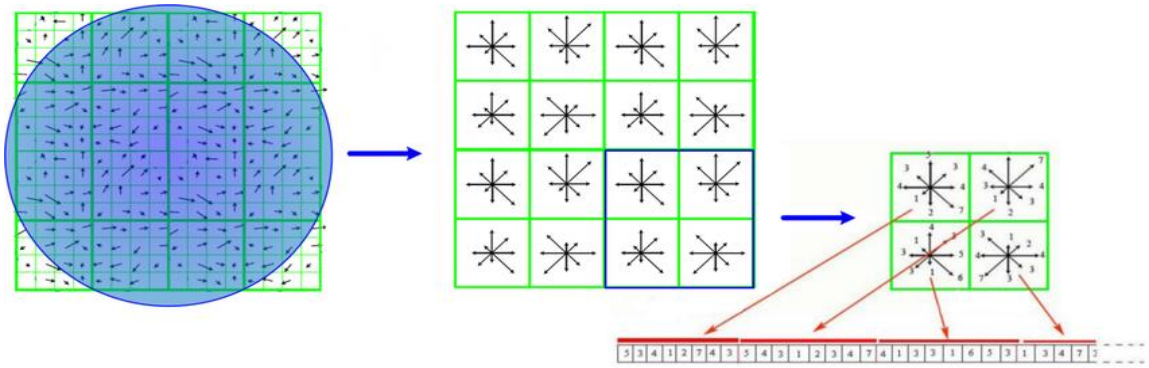

Figura 12. Característica y descriptor SIFT.

Fuente:(Doctoral, 2015)

Para realizar el seguimiento es decir la correspondencia de los descriptores en la secuencia de las imágenes se tiene diversos métodos de emparejamiento o *matching* para esto se toma encuentra una medida de similitud, para hacer mención en los casos de los descriptores SSD o NCC se usa la luminosidad mientras que en los descriptores como SIFT se usan la distancia euclidiana.(Doctoral, 2015)

Una vez realizado la etapa de describir las características de la imagen se las compara de tal forma que se acojan para cada descriptor de la primera imagen el mejor descriptor de la segunda. Puede suceder que a veces un descriptor de la segunda imagen quede vinculado a la mas de un descriptor de la primera , para esto se puede realizar comparaciones inversas que un descriptor de la primera imagen que vinculado a varios de la segunda pero el coste computacional es alto y mucho más cuando se trata de realizarlo en tiempo real por lo que se opta realizar búsquedas en regiones donde se asume que aparecerán los descriptores , esto recae en la generación de modelos predictivos los cuales están basados en sensores como IMU , GPS, laser entre otros.(Fraundorfer & Scaramuzza, 2012)

2.4.Flujo óptico

El seguimiento de objetos tiene muchas aplicaciones como sistemas autónomos de navegación para automóviles o sistemas de vigilancia (Yamamoto, Mae, Shirai, & Miura, n.d.). Un método usado para el seguimiento de objetos dentro de una escena es el tracking basado en la extracción del flujo óptico. El flujo óptico es el patrón de movimiento aparente de un objeto, superficie o bordes, entre dos cuadros de imagen consecutivos, debido al movimiento de la cámara o un objeto dentro de la escena (Patel, 2013).

La técnica consiste en calcular el campo vectorial donde cada vector es un vector de desplazamiento que muestra el movimiento de los puntos característicos entre dos frames consecutivos. Estos campos de desplazamiento se calculan de forma local en el entorno de un punto característico. La **Figura 13** muestra la representación de los vectores de desplazamiento en el plano de una imagen producido por un giro del observador.

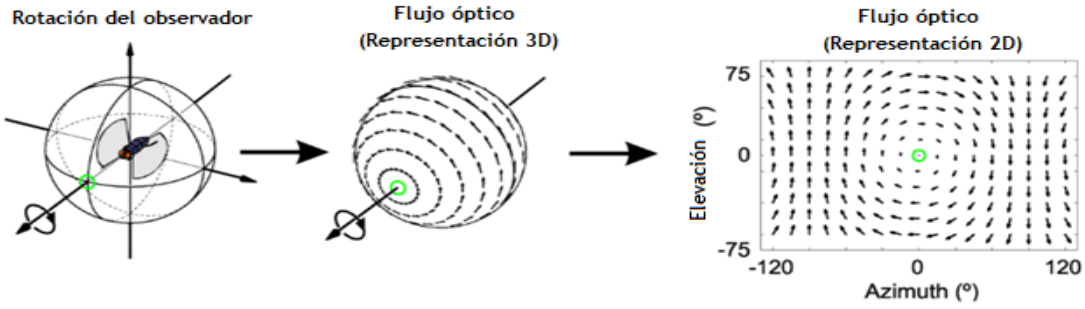

Figura 13.Flujo óptico producido por el giro del observador.

Fuente: Optical flow (Patel, 2013)

La **Figura 14** muestra la relación que existe en el vector de desplazamiento de un objeto en el mundo real y el vector de flujo óptico en el plano de imagen. Es por eso que el campo de flujo óptico se define como una proyección del movimiento de un objeto en el espacio.

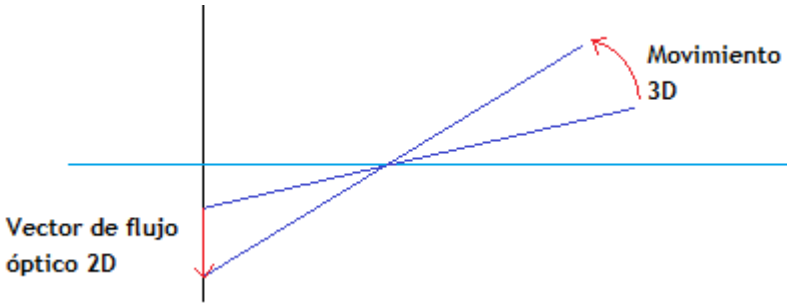

Figura 14.Relación entre el movimiento en 3D y el movimiento en el plano de la imagen.

Fuente: Optical Flow (Patel, 2013)

Matemáticamente se define de la siguiente forma: asumimos un pixel $I(x, y)$ que se desplaza una distancia (p, q) en las direcciones x y y respectivamente. Si la intensidad del pixel no cambia, este pude ser localizado en la siguiente imagen mediante la siguiente ecuación:

$$H(x, y) = I(x + p, y + q) \tag{29}$$

$$0 = I(x + p, y + q) - H(x, y)$$

$$0 \approx I(x, y) + I_x p + I_y q - H(x, y)$$

$$0 \approx (I(x, y) - H(x, y)) + I_x p + I_y q$$

$$0 \approx I_t + I_x p + I_y q$$

$$0 \approx I_t \nabla I \cdot [p, q] \tag{30}$$

Donde I_x, I_y, I_t son las derivadas parciales y $p = \frac{\partial x}{\partial t}$, $q = \frac{\partial y}{\partial t}$.

El modelo de movimiento del flujo óptico en 2D asume que $I(x, y)$ es el centro de una pequeña región de $n \times n$, denominada también vecindario, que se mueve δ_x, δ_y en un tiempo δ_t a un punto $(x + \delta_x, y + \delta_y, t + \delta_t)$, tenemos

$$I(x, y, t) = I\left(x + \delta_x, y + \delta_y, t + \delta_t\right) \quad (31)$$

Los movimientos provistos por δ son muy pequeños, de modo que una serie de Taylor de primer orden se puede usar (Patel, 2013) para obtener

$$I\left(x + \delta_x, y + \delta_y, t + \delta_t\right) = I(x, y, t) + \frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\,\delta_y + \frac{\partial I}{\partial t}\delta_t + H.O.T \quad (32)$$

Donde H.O.T so los términos de alto orden que se asumen como cero pues son demasiado pequeños. Reemplazando (31) en (32) se obtiene:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}\,v_y + \frac{\partial I}{\partial t}v_t = 0 \quad (33)$$

Donde $v_x = \frac{\partial x}{\partial t}$ y $v_y = \frac{\partial y}{\partial t}$ son los componentes de la velocidad (flujo óptico) y $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$, $I_t = \frac{\partial I}{\partial t}$ son las derivadas parciales de la intensidad de la imagen en esa región. Reescribiendo la ecuación:

$$\left(I_x, I_y\right) \cdot \left(V_x, V_y\right) = -I_t$$

$$\nabla I \cdot \breve{v} = -I_t \quad (34)$$

Donde $\nabla I \cdot \breve{v} = -I_t$ se conoce como movimiento 2D y $\breve{v}$ es el flujo óptico en un píxel.

- **Método de Lucas-Kanade para medir el flujo óptico**

Este método tipo diferencial resuelve las ecuaciones básicas del flujo óptico para todos los píxeles de un vecindario mediante el criterio de mínimos cuadrados. Sin embargo este método no permite calcular el flujo óptico en regiones uniformes de una imagen, debido a que se trata de un método puramente local (Patel, 2013).

El método de Lucas-Kanade asume un desplazamiento pequeño y constante dentro de un vecindario de un pixel p entre dos frames consecutivos. Además se asume que el pixel p es un punto central dentro de una ventana (Patel, 2013). De esta forma tenemos:

$$I_x(p_1)V_x + I_y(p_1)V_y = -I_t(p_1)$$

$$I_x(p_2)V_x + I_y(p_2)V_y = -I_t(p_2)$$

$$\vdots$$

$$I_x(p_n)V_x + I_y(p_n)V_y = -I_t(p_n)$$

Donde $p_1, p_2 \dots p_n$ son los píxeles dentro de la ventana. $I_x(p_n), I_y(p_n), I_t(p_n)$ Son las derivadas parciales de la imagen respecto a la posición (x, y) y el instante t, evaluadas en el punto p_n. Estas ecuaciones escritas matricialmente:

$$Av = b \text{ Donde} \qquad (35)$$

$$A\begin{bmatrix} I_x(p1) & I_y(p1) \\ I_x(p2) & I_y(p2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_3) \end{bmatrix}$$

Aplicando el criterio de mínimos cuadrados, esta ecuación queda resuelta. Sin embargo pueden presentarse errores en el método, cuando no se satisface la constancia de brillo en el pixel de un

frame a otro, cuando el movimiento no siempre es pequeño, cuando el tamaño de la venta es demasiado pequeño (Patel, 2013).

2.4.1. Algoritmo RANSAC

El conjunto de correspondencias calculado mediante los métodos de matching, tracking o flujo óptico, tienen errores en la asociación denominados como valores anómalos (*outliers*). Debido al ruido en imágenes, oclusión, desenfoque, cambios de punto e visión e iluminación. Para que la estimación sea más precisa es conveniente eliminar estos valores. Existen un método basado en las restricciones geométricas del modelo de movimiento RANSAC.(Doctoral, 2015)(W. G. Aguilar & Angulo, 2015)

RANSAC trata de obtener un modelo de hipótesis a partir de un conjunto de puntos y después verificando el modelo con otros puntos de la imagen, la hipótesis que tenga mejor aceptación con el conjunto de datos será la solución.(Fraundorfer & Scaramuzza, 2012) El número de subconjuntos a analizar depende del número de iteraciones que sea definido esto influye en la precisión del resultado final, entonces se dice que RANSAC es un método probabilístico es decir que la solución no va hacer la misma en cada una de la ejecuciones, a pesar de ello entre más repeticiones se le dé el resultado final será mejor(Lacey et al., 2002). De aquí se parte con dos criterios para optimizar el código y coste computacional ya sea que se disminuya el número d puntos de interés o se reduzca el número de iteraciones.(Lacey et al., 2002)

Algoritmo1: RANSAC

1: Selecciona el número mínimo de puntos, útiles para generar el modelo de parámetros

2: Resolución de los parámetros en el modelo.

3: Define el número de puntos del conjunto que encajan según la tolerancia definida.

4: SI el número de puntos que encajan sobre el número total de puntos en el conjunto supera el umbral, vuelve a calcular los parámetros del modelo con los puntos que encajan.

5: Caso contrario, repetir paso 1 a 4, N veces.

Para elegir el número de iteraciones N se considera que debe ser lo suficientemente alto para garantizar la probabilidad p(establecida en un 0.99).(Derpanis, 2010)

El número de iteraciones viene expresado por:

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^S)} \quad (36)$$

N: número de iteraciones

S: Número mínimo de puntos de interés

ϵ: Porcentaje de valores anómalos esperados

p: probabilidad de éxito requerida

2.5.Unidad de medidas inerciales (IMU)

La navegación inercial es una técnica que parte de la medición proporcionados por acelerómetro y giroscopios, los cuales se usan para rastrear posición y orientación de un objeto referente a un punto, orientación y velocidad conocida.(Armesto, Tornero, & Vincze, 2007)

Las unidades de medidas inerciales (IMU), típicamente suelen estar conformados por tres giroscopios y tres acelerómetros ortogonalmente dispuesto, estos miden la velocidad angular y aceleración lineal respectivamente. Cuando se procesa las señales de estos dispositivos se puede rastrear la posición y orientación del objeto.(Guallichico & Ávalos, 2013)

Las aplicaciones son variadas desde navegación aeronáutica, misiles estratégicos, naves espaciales y submarinos. La creación de dispositivos inercialmente pequeños y ligeros ha hecho posible ampliar la gama de aplicaciones, incursionando en áreas de movimiento humano y comportamiento animal.(Woodman, 2007)

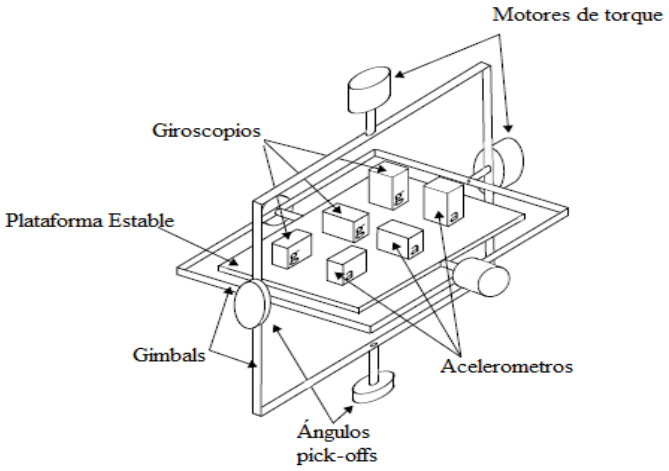

Figura 15. Sistema de plataforma estale

Fuente: .(Armesto et al., 2007)

En un sistema de plataforma estable, para seguir la orientación del dispositivo se mide los ángulos de cardanes (*gimbals)* adyacentes a partir de los ángulos recogidos (*pick-offs*).

Para el cálculo de la posición del dispositivo las señales del acelerómetro deben ser doblemente integradas, sin embargo, se debe tener en cuenta que los valores contienen el valor de aceleración de la gravedad en vertical por lo que es necesario restar dicho valor en cada canal antes de realizar la doble integración.(Woodman, 2007)

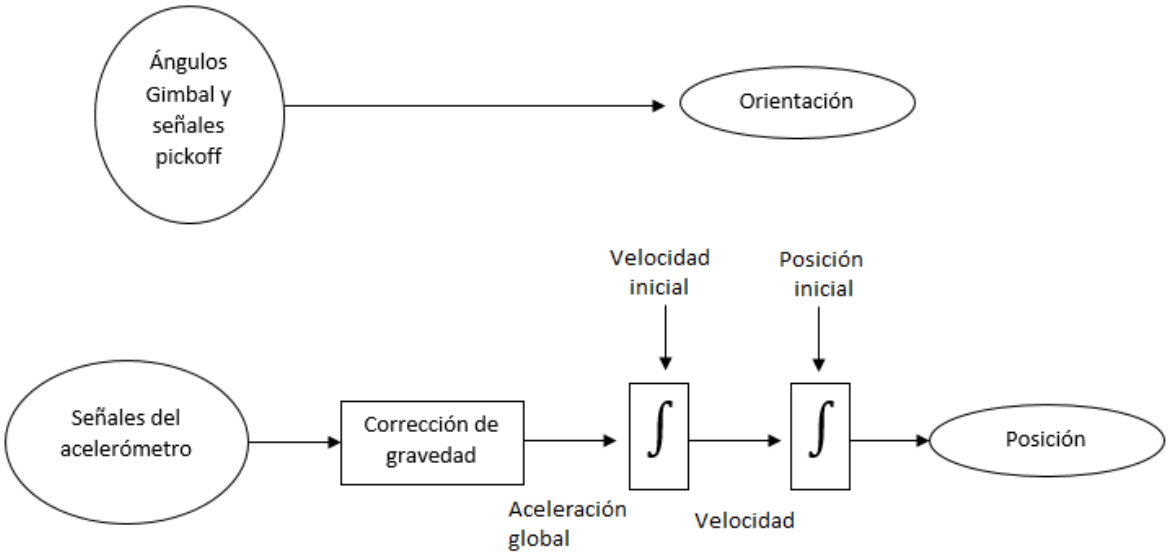

Figura 16. Esquema para la obtención de orientación y posición.

Fuente: (Woodman, 2007)

2.5.1. Seguimiento de orientación

La orientación o la actitud(*attitude*) es rastreada por la integración de la señal de velocidad angular $w_b(t) = (w_{bx}(t), w_{by}(t), w_{bz}(t))^T$, como es evidente esta conformado por los tres del giroscopio, para expresar la orientación es necesario tener varias representación de *attitude* como los ángulos de Euler , cuaterniones y cosenos directores.(Guallichico & Ávalos, 2013)

Si representamos en función de cosenos tenemos que la *attitude* esta descrita en una matriz C de 3x3, donde cada columna representa a un vector unitario de cada eje del cuerpo.

Un vector v_b definido en el marco del cuerpo es equivalente a:

$$v_g = C v_b \tag{37}$$

Definido el marco global la transpuesta estada dada por:

$$v_b = C^T v_g \tag{38}$$

Esto resulta ya que el inverso de una matriz de rotación es igual a su transpuesta. Para rastrear la *attitude* se debe rastrear la matriz C a lo largo del tiempo es decir que la *attitude* viene expresada por:

$$\dot{C(t)} = \lim_{\delta t \to 0} \frac{C(t+\delta t) - C(t)}{\delta t} \tag{39}$$

$C(t+\delta t)$ se puede expresar como el producto de dos matrices tal como:

$$C(t+\delta t) = C(t)A(t) \tag{40}$$

A(t) es la matriz de rotación que relación el marco del cuerpo en in instante t y $t+\delta t$. Si $\delta\phi, \delta\theta\ y\ \delta\psi$ son rotaciones pequeñas del cuerpo en los tiempos t y $t+\delta t$ en el eje x, eje y y eje z respectivamente se puede usar una aproximación descrita como:

$$A(t) = I + \delta\Psi \tag{41}$$

Dónde:

$$\delta\Psi = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{pmatrix} \tag{42}$$

Entonces sustituyendo:

$$C(t) = \lim_{\delta t \to 0} \frac{C(t+\delta t) - C(t)}{\delta t}$$

$$C(t) = \lim_{\delta t \to 0} \frac{C(t)A(t) - C(t)}{\delta t}$$

$$\dot{C(t)} = \lim_{\delta t \to 0} \frac{C(t)(I+\delta\Psi) - C(t)}{\delta t} \qquad (43)$$

$$\dot{C(t)} = C(t) \lim_{\delta t \to 0} \frac{\delta\Psi}{\delta t}$$

Si el límite del ángulo pequeño es un valor aceptable se puede expresar lo siguiente:

$$\lim_{\delta t \to 0} \frac{\delta\Psi}{\delta t} = \Omega(t) \qquad (44)$$

Donde

$$\Omega(t) = \begin{pmatrix} 0 & -w_{bz}(t) & w_{by}(t) \\ w_{bz}(t) & 0 & -w_{bx}(t) \\ -w_{by}(t) & w_{bx}(t) & 0 \end{pmatrix}$$

Esta es la forma simétrica del vector angular wb(t). entonces para continuar con el seguimiento de la orientación hay que resolver la ecuación diferencial.(Woodman, 2007)

$$\dot{C(t)} = C(t)\Omega(t) \qquad (45)$$

La solución está dada por la siguiente expresión:

$$C(t) = C(0)\exp(\int_0^t \Omega(t)dt) \qquad (46)$$

C(0) es la orientación inicial del dispositivo.

2.5.2. Seguimiento de posición

La señal de aceleración está dada por $a_b(t) = (a_{bx}(t), a_{by}(t), a_{bz}(t))^T$, los acelerómetros proyectan el maro en referencia global tal como:

$$a_g(t) = C(t)a_b(t) \quad (47)$$

Debido a la aceleración que esta implícitamente en los datos por causa de la gravedad se le resta y una vez obtenido los nuevos valores de aceleraciones se integran dos veces, la primera con el fin de encontrar la velocidad y la segunda para obtener el desplazamiento.

$$v_g(t) = v_g(0) + \int_0^t a_g(t) - g_g dt \quad (48)$$

$$s_g(t) = s_g(0) + \int_0^t v_g(t) dt \quad (49)$$

Donde $v_g(0)$ y $s_g(0)$ corresponden a velocidad inicial y desplazamiento inicial del dispositivo.

Para llevar acabo lo descrito anteriormente por las ecuaciones se debe utilizar un esquema de integración, y una implementación simple es usar una regla rectangular y usa la siguiente actualización en las ecuaciones.

$$v_g(t + \delta t) = v_g(t) + \delta t(a_g(t + \delta t) - g_g) \quad (50)$$

$$s_g(t + \delta t) = s_g(t) + \delta t\, v_g(t + \delta t) \quad (51)$$

Sin embargo no es el único método para realizar la doble integración ya que algunos métodos toman más en serio la propagación del error con otros criterios u otros métodos se han desarrollado técnicas de integración numérica tal como la integración mediante punto medio (RK2) o la integración por Runge-Kutta4 (RK4) ,siendo usados frecuentemente en la literatura usado en la literatura .(Woodman, 2007)

2.5.3. Cuaterniones

Los cuaterniones fueron un invento de Hamilton y fueron descritos como objetos matemáticos llamados cuaterniones (*Quaternions*)(A, 2014).Debido a su complejidad no han sido usados ampliamente sin embargo en la ingeniería informática se le ha dado uso para representar rotaciones en un espacio 3D .(Markelov, 2002)(En, Atica, & De, n.d.)

Un cuaternio se puede definir como un objeto de cuatro dimensiones

$$q = w + xi + yj + zk \quad (52)$$

Dónde:

w, x, y, z Son números reales, por lo que se puede asumir que son elementos del espacio $R^4 = RxRxRxR$, por otro lado i, j, k son unidades imaginarias.(En et al., n.d.)

2.5.3.1.Rotaciones con cuaterniones

Un punto en el espacio es representado con cuaterniones de parte real nula y se los denomina cuaterniones puros(“Cap ´ Rotaciones y Cuaternios,” n.d.). Los cuaterniones unitarios representan rotaciones de objetos en tres dimensiones y se los puede denotar como $q = (\cos\left(\frac{\alpha}{2}\right), v'sen(\frac{\alpha}{2}))$ siendo v' un vector en R^3. Por lo que todo cuaternio unitario posee un ángulo y una dirección y es asociable a una rotación alrededor del origen.(En et al., n.d.)

Entonces siendo un punto P en el espacio (cuaternion puro), la transformación se puede expresar de la siguiente forma:

$$T_q(P) = q.P.q^* \quad (53)$$

Como resultado tenemos $P' = T_q(P)$ que es la imagen de P por su rotación en el eje v' un ángulo α

2.5.3.2.Rotación con matrices de Euler

Es uno de los métodos más usado cuando se trata de representar rotaciones, esto se logra mediante la construcción de matrices de rotación básicas sobre los ejes cartesianos, de tal forma que $R = R_{(Z,\gamma)}.R_{(Y,\beta)}.R_{(X,\alpha)}$ de un ángulo α alrededor del eje X, seguido de una rotación de un ángulo β alrededor del eje Y y por ultimo un rotación con un ángulo γ alrededor del eje Z.(A, 2014)

$$R_{(X,\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix}; R_{(Y,\beta)} = \begin{bmatrix} \cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix};$$

$$R_{(Z,\gamma)} = \begin{bmatrix} \cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Esto representa un giro en sentido anti horario del sistema de referencias.

Pero existen una desventaja considerable y es la aparición de singularidades debido al *Gimbal lock* por lo que este método queda descartado para aplicación automáticas.(“Cap ´ Rotaciones y Cuaternios,” n.d.)

2.5.3.3 Problema del Gimbal lock

Los cuaterniones son usados en el campo de la computación específicamente en animación de objetos en 3D orientándose a la creación de videojuegos y es ahí donde la geometría cobra

importancia, por lo que son considerados como una herramienta matemática importante. Lo que hacen los cuaternios en términos más simples se encargan de las rotaciones de un objeto y se diferencia de los ángulos de Euler los mismo que para producir una rotación los cardanes mueve todos a la vez provocando en ocasiones un suceso denominado *Gimbal lock* que es representado cuando que los ejes de giro se sobreponen tomando valores distintos en ese instante causando así un giro no planeado o bloqueando el giro .Los ángulos de Euler describen trayectorias o giros innecesarios para llegar a una orientación definida esto se debe al movimiento de los tres ejes . (Markelov, 2002)

Estos inconvenientes mencionados anteriormente son solucionados en gran medida por los cuaterniones ya que para obtener una orientación son más eficientes y lo realizan girando de forma más uniforme evitando trayectorias innecesarias por lo que son más rápidos.(En et al., n.d.)

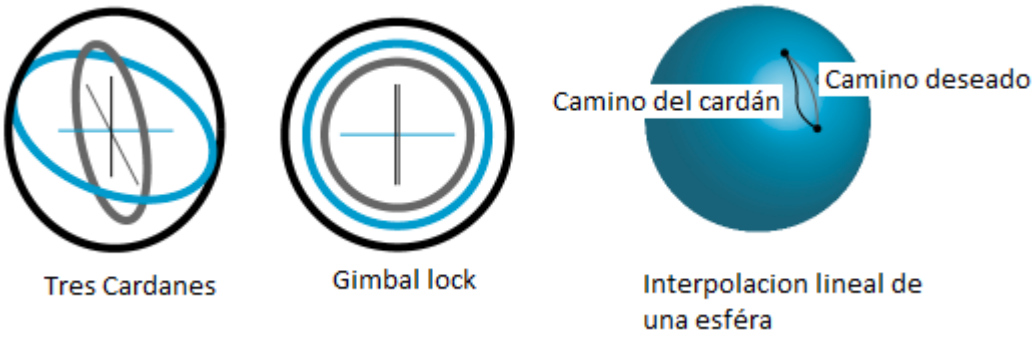

Figura 17. Representación del Gimbal lock.

Fuente: (Markelov, 2002)

2.5.4. Integración numérica método Runge-Kutta

Para comprender el uso de la integración por Runge-Kutta, partiremos del cálculo de la orientación y una forma de implementarlo(D. Scaramuzza, Fraundorfer, & Siegwart, 2009). La actitud o *attitude* muestra la inclinación del dispositivo con respecto al eje longitudinal y lateral. la orientación por otro lao no permite conocer el rumbo de navegación del objeto con respecto al norte magnético, este sistema es conocido como AHRS(*Attitude and Heading Reference System*). Está representado por los ángulos de euler denominados *roll,pitch y yaw*.(Guallichico & Ávalos, 2013)

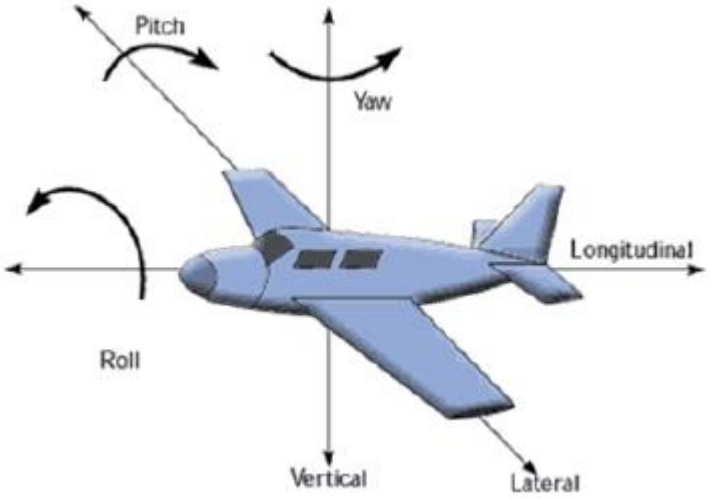

Figura 18. Sistema AHRS

Fuente: (Guallichico & Ávalos, 2013)

Para conocer la actitud es necesario calcular los ángulos de Euler *roll,pitch y yaw* ,por lo tanto es necesario resolver las ecuaciones siguientes.

$$\dot{E} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi sec\theta & cos\phi sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (54)$$

Donde

$\begin{bmatrix} p \\ q \\ r \end{bmatrix}$ es el vector de velocidades angulares medidos por los giroscopios en el eje x,y y z .

$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$ es el vector de la derivadas de los ángulos *roll,pitch y yaw*

El sistema muestra un conjunto de ecuaciones diferenciales discretas y para resolverlas se puede usar Rugen-Kutta que esta descrita por la siguiente expresión:

$$y_{i+1} = y_i + \frac{(k1 + 2 * k2 + 2 * k3 + k4)}{6} * h \quad (55)$$

Dónde:

$$k1 = f(x_i, y_i)$$

$$k2 = f(x_i + \frac{1}{2} * h, y_i + \frac{1}{2} * h * k1)$$

$$k3 = f(x_i + \frac{1}{2} * h, y_i + \frac{1}{2} * h * k2)$$

$$k4 = f(x_i + h, y_i + h * k3)$$

$$h = tiempo\ de\ muestreo$$

$$x_i = \begin{bmatrix} x0 \\ x1 \\ x2 \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$y_i = \begin{bmatrix} y0 \\ y1 \\ y2 \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

$$h = dt = 0.01$$

Por lo tanto:

$$f_i = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_0 & x_1 siny_0 tany_1 & x_2 cosy_0 tany_1 \\ 0 & x_1 cosy_0 & -x_2 siny_0 \\ 0 & x_1 siny_0 secy_1 & x_2 cosy_0 secy_1 \end{bmatrix} \quad (56)$$

De esta manera se calcula los ángulos de Euler a partir de varias iteraciones en un tiempo de muestreo determinado, el RK calcula la actitud a partir del valor anterior por lo que genera errores acumulativos.(Guallichico & Ávalos, 2013)

2.5.5. Altímetro

Para la medición de la altura se debe partir de la medición de la presión atmosférica esto se logra mediante los barómetros, estos sensores miden la presión del aire. Existen diferentes tipos de barómetro como Torricellan y aneroid que viene siendo los barómetros mecánicos tradicionales, sin embargo, en nuestro tiempo ya podemos contar con sensores digitales y mejor aún ya integrados en los teléfonos inteligentes.(Asociados, World Meteorological Organization, & Wmo, 2005)

Los fabricantes de teléfonos inteligentes incluyen este sensor con el fin de mejorar la localización con conjunto con el GPS, ya que se puede conocer a partir de la presión medida la altura en la que se encuentra.

Otra aplicación del barómetro es predecir el estado climático, conforme la presión suba o baje se indica cambios en el clima, tal cual si este sube el clima será bueno mientras este disminuya su valor es más probable que llueva, nieve según la localización.(Asociados et al., 2005)

Para conocer la altura según el barómetro se debe emplear una fórmula matemática que esta embebida en los dispositivos :

$$Altitud = 44330 * \left(1 - \frac{Presión}{Presion\ atmosferica\ estandar}^{Coeficiente\ de\ altitud}\right)[m] \quad (57)$$

Dónde:

Presión: =datos obtenidos del barómetro [hPa]

Presión atmosférica estándar =1013.25 [hPa]

Coeficiente de altitud= $\frac{1}{5.2555}$

Constante de conversión= 44330[m]

2.6.Software

En esta sección se describen las plataformas y lenguajes de programación utilizados en este proyecto de investigación.

2.6.1. Unity

Unity es un motor de desarrollo para la creación de videos juegos y contenido multimedia con animaciones 3D, brinda facilidades para la creación de aplicaciones en diversas plataformas desde dispositivos móviles hasta equipos de sobre mesa. Unity posee un lenguaje de programación C# el cual posee su editor propio o a su vez se puede a hacer uso en Visual Studio para compilar los scripts, los cuales son los archivos donde se plasma la programación para luego ser agregados a los objetos en 3D.(Under & Control, n.d.)

Figura 19. Unity 3D

Fuente: Unity documentation(Under & Control, n.d.)

2.6.1.1.OpenCV en Unity

OpenCV es una librería que contiene algoritmos de visión por computadora, inicialmente estaba escrita y optimizada para C sin embargo por sus prestaciones se vio conveniente extrapolar a diversos leguajes de programación tal como C++, Python, Matlab, Ruby y hasta C# lenguaje de programación en Unity. (“About - OpenCV library,” n.d.)

La esencia de esta librería es ofrecer a los desarrolladores una herramienta de programación con el fin que sus aplicaciones sean más sofisticadas y conlleven menos tiempo más aún si se trata de la creación de aplicaciones en el campo de visión por computador.

OpenCV contiene más de 500 funciones que abarcan varias áreas donde se aplica visión por computadora tales como en inspección de productos en fábricas, escaneos médicos, seguridad, interfaz de usuarios, calibración de cámara, robótica entre otras más.

La integración de OpenCv a Unity se da mediante la tienda de adquisidor de *assets* ubicada en el mismo Unity ahí se puede encontrar diversos paquetes descargables para complementar los proyectos en los que se estén trabajando. La librería está en el apartado de *assets* de paga, sin embrago la adquisición de la librería no es completa debido a que se ha modificado para el

entendimiento del lenguaje que maneja Unity por lo que no contiene todas las funciones que han sido implementadas para lenguajes como C o C++.

Figura 20. OpenCV para Unity.

Fuente: Unity Asset Store

2.6.1.2.Android SDK para Unity

El sistema operativo Android para móviles era prácticamente desconocido hasta que en 2005 donde Google lo adquirió. En 2007 específicamente en el mes de noviembre se creó la Open Handset Alliance, que agrupó a muchos fabricantes de teléfonos móviles y procesadores. Este año se lanzó la primera versión de Android, junto con el SDK (del inglés, Software Development Kit, que significa Kit del desarrollo de software).(Mentor & Programaci, n.d.)

El SDK utiliza una estructura modular que separa las distintas versiones de Android, complementos, herramientas, ejemplos y la documentación en un único paquete que se puede instalar por separado. Para desarrollar una aplicación en Android, es necesario descargar, al menos, una versión.(Under & Control, n.d.)

Depende del dispositivo la versión de software para el cual se dese construir la aplicación, a pesar de tener soporte desde Android 2.3.6 en adelante, Unity también tiene sus limitaciones debido a que también muestra continuamente actualizaciones de versión por lo que podemos diferenciar que para las versiones 4.x.x tenia soporte para sistemas que cuentan con Android 2.3.6 mientras que en la versiones recientes de Unity 5.x.x tiene soporte a partir de Android 4.4

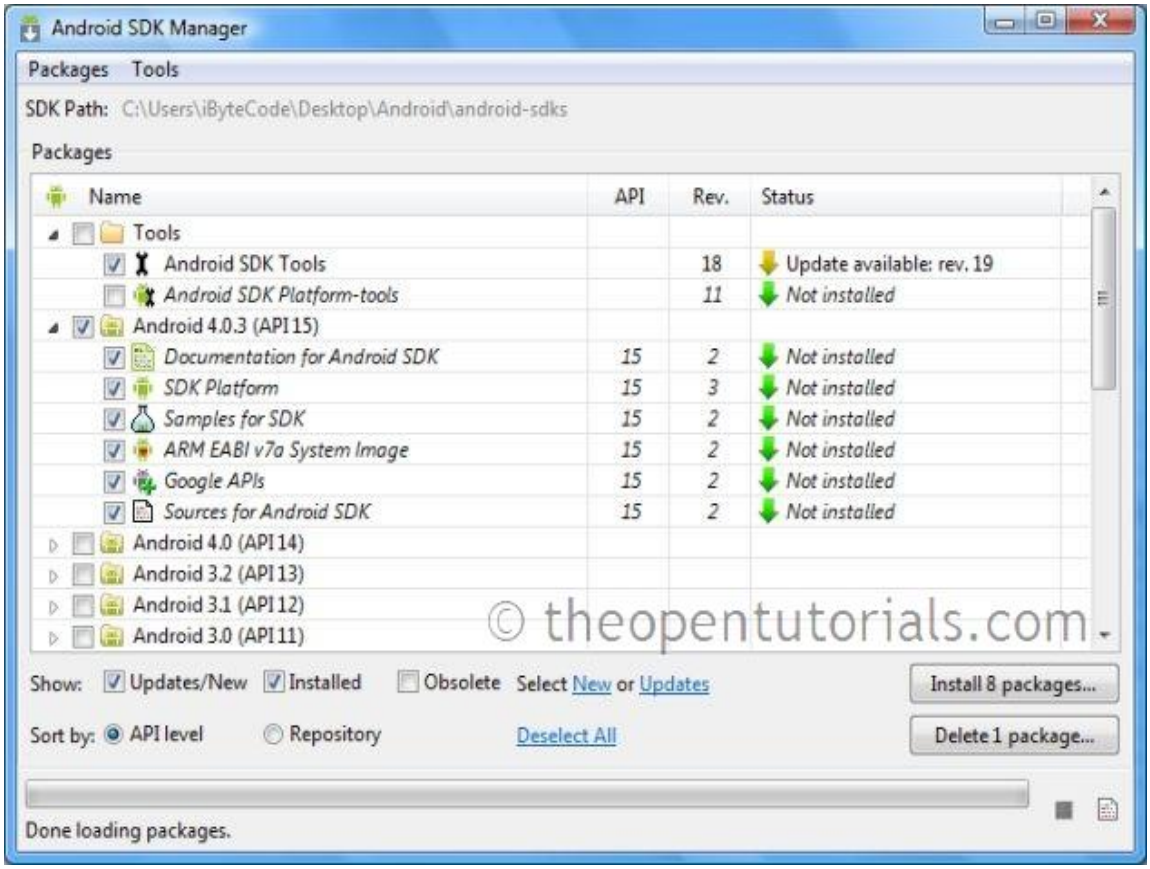

Figura 21. Instalación de paquetes de desarrollador.

2.6.1.3.Manejo de Unity

Existen dos versiones de uso para la plataforma una es la versión gratuita donde se va ver limitación en el monto de realizar aplicación para ciertas plataformas y la versión profesional donde se puede desarrollar sin problemas para todos las plataformas que soporta Unity.

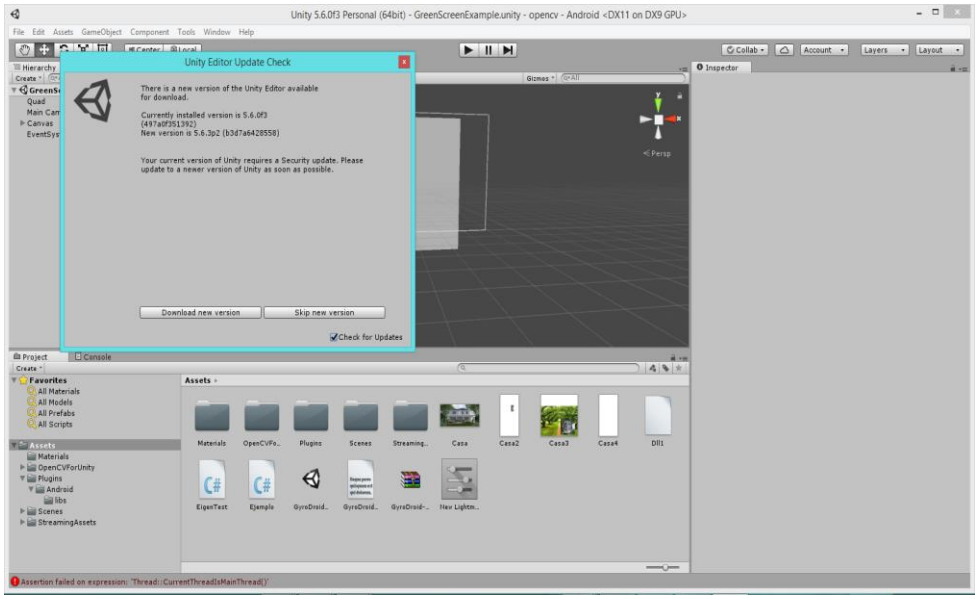

Figura 22. Acceso a Unity.

Con lo que corresponde a la interfaz de usuario hay que localizar las áreas de trabajos y herramientas que estarán disponibles dependiendo cual sea la necesidad del proyecto en el cual se esté trabajando.

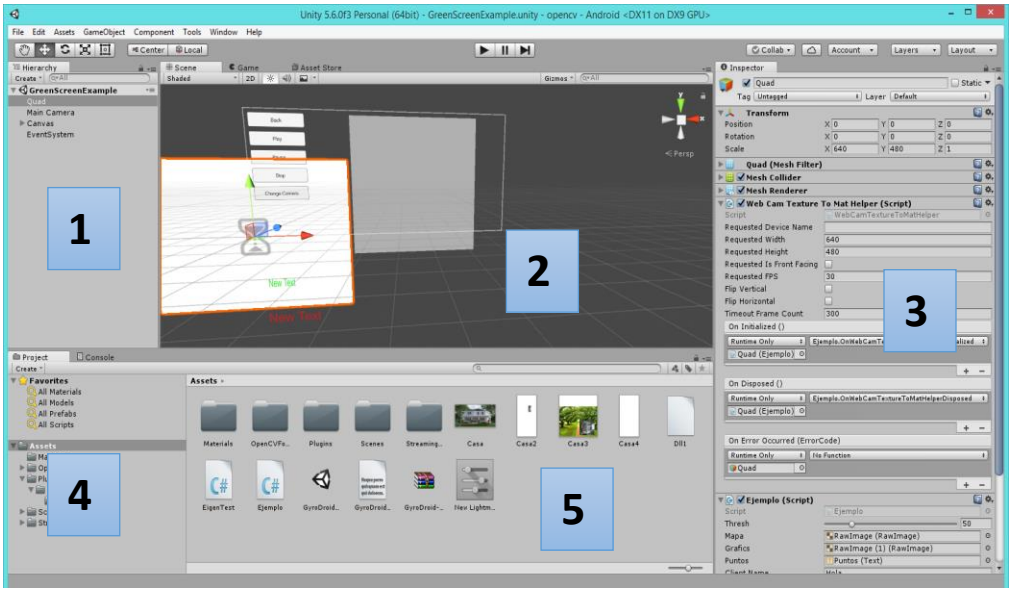

Figura 23. Interfaz de Usuario

1. Vista de jerarquía donde se alojan los objetos dentro de la escena, librerías y *scripts*
2. Vista de escena donde se va desarrollando la parte grafica del proyecto. La forma más simple de ir creando un entorno es arrastrando los elementos a los cuales se los podrá modificar sus atributos directamente como rotarlos, aumentar escala y posicionarlos.
3. Vista de inspector este es un menú que despliega los atributos de los objetos seleccionados es otra forma más precisa de modificar los objetos, y que además se puede ir agregando más atributos de color texturas hasta scripts donde se les dota a los objetos de animaciones o de cierta inteligencia para que interactuasen dentro del entorno de desarrollo.
4. Vista de proyecto, aquí se alojan los componentes de todo el proyecto como librerías, objetos 3D, scripts entre otros. Aquí es donde se va a ir agregando *assets* o paquete de complemento para el proyecto ya sea con una descarga directa de la tienda o arrastrando los paquetes en este apartado.
5. Vista del buscador es útil para ubicar objetos o archivos que se encuentren dentro del proyecto de una forma más rápida.

- **Menú de aplicaciones**

Está ubicado en la parte superior de la pantalla, y es aquí donde se podrá manipular a nivel de proyecto lo que se desea hacer como guardar, abrir otra escena o proyecto hasta la creación de las aplicaciones en función de la plataforma de elección.

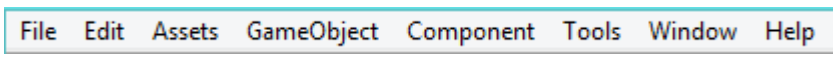

Figura 24. Menú de aplicaciones

- **Botones de control de objetos de la escena**

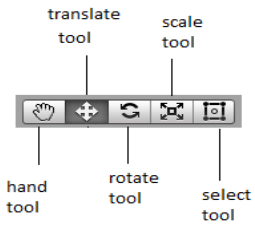

Figura 25. Botones de control

Hand tool : con este icono nos podemos mover dentro de la escena con el fin de ubicar los elementos o una región de interés.

Translate tool: sirve para mover cualquier objeto dentro de la escena en los ejes X,Y, Z.

Rotate tool: nos permite rotar cualquier objeto que este dentro de la escena.

Scale tool: nos permite modificar la escala de un objeto es decir su tamaño se verá afectado con respecto a los demás objetos dentro de la escena.

Select tool: si se trata de un objeto que pose esquinas nos permite seleccionar la esquinas con el fin de ir modificándolo.

- **Botones de ejecución**

Figura 26. Botones de ejecución

Unity es capaz de ejecutar cualquier aplicación ante ser construida, lo cual es favorable debido a que se da vistazo previo dando a conocer la ejecución de la aplicación, aquí puede aún detectar fallos en el entorno visual o de programación.

Estos botones están ubicados en la parte superior de la pantalla centrados y cuenta con un botón de inicio para ejecutar la aplicación, uno de pause y un botón de avance.

- **Scripts**

Los scripts permiten la definición de la lógica del juego. El funcionamiento es muy sencillo, primero se define el comportamiento deseado en un script y este se agrega como un componente al objeto. Permite la gestión de eventos (activar/desactivar), valores de entrada por parte del dispositivo sobre el cual el programa se ejecuta ("Unity - Manual: Creando y usando scripts," n.d.), como son teclas, mouse, acelerómetro, giroscopio, cámara y demás sensores que este posee. La ***Figura 27*** muestra el orden de ejecución de las líneas de código dentro de un Script.

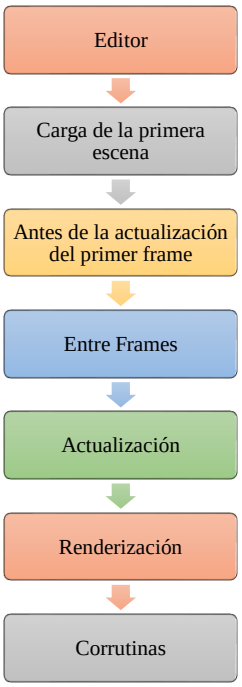

Figura 27. Orden de ejecución de funciones

Fuente: Orden de ejecución ("Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento)," n.d.)

En la sección del editor dentro de un script de Unity se encuentra la función Reset. Esta función se ejecuta para inicializar las propiedades del script (“Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento),” n.d.).

Carga de la primera escena, en esta sección se encuentran funciones como Awake, OnEnable. La función Awake se ejecuta antes del Start, es útil para inicializar variables o estados del programa antes que inicie (“Unity - Scripting API: MonoBehaviour.Awake(),” n.d.).

Antes de la actualización del primer frame se ejecuta la función Start, justo después de la función Awake. De la misma forma que la función Awake, Start se utiliza para inicializar variables y se ejecuta una sola vez durante la vida del Script (“Unity - Scripting API: MonoBehaviour.Start(),” n.d.).

La sección entre frames contiene la función OnApplicationPause, esta se ejecuta al final del frame donde la pausa se ejecutó. Lugo que OnApplicationPause ha sido llamada, se ejecuta un frame extra con la finalidad de ejecutar código que indique que el programa esta pausado (“Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento),” n.d.).

La mayoría de las tareas del Script se ejecutan dentro de la función Update que se encuentra en la sección de Actualización. Update se llama una vez por frame(“Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento),” n.d.). Y es la función principal para las actualizaciones de frames (“Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento),” n.d.). Sin embargo, en esta sección existen otras funciones que se ejecutan en esta sección como FixedUpdate, útil cuando se requiere una taza de actualización más veloz que la predeterminada.

Una función importante dentro de la programación de Scripts es OnGUI. Se utiliza para leer eventos por parte del usuario ("Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento)," n.d.) como pausas, inicio, stop, mouse, entre otras. Esta función se encuentra en la sección de renderización junto a otras funciones. OnGUI se ejecuta varias veces por frame ("Unity - Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento)," n.d.) con la finalidad de captar los eventos producidos por parte del usuario.

Finalmente, dentro de la estructura de ejecución de un Script dentro de Unity, están las Corrutinas. Este tipo de funciones que se ejecutan de forma paralela y tienen la ventaja de suspender su ejecución en cualquier momento una vez que el comando Yield ha sido ejecutado en el siguiente frame. Además, Unity permite la ejecución de Threads, funciones similares a las corrutinas, con la única diferencia que estas no pueden ser pausadas.

Unity 3D posee la clase Input, esta permite capturar datos de dispositivos de entrada como joypads, teclados, mouse, controles. La clase Input permite trabajar además con datos de pantallas táctiles y sensores de movimiento inercial disponibles en dispositivos móviles. Esta clase permite a Unity hacer uso de dispositivos de audio y video como cámaras, micrófonos presentes en el dispositivo sobre el que se ejecuta el programa ("Unity - Manual: Input," n.d.).

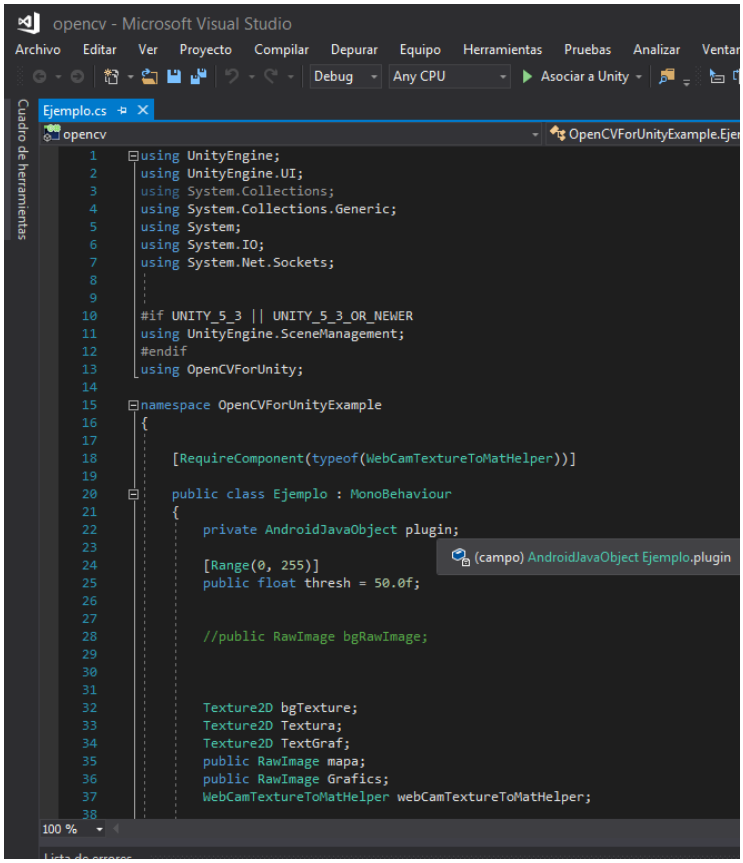

Figura 28. Ejemplo de un Script en Visual Studio.

2.6.1.4.Creación de la aplicación

Finalizado el proyecto, se compila la aplicación para poder usarla desde un teléfono móvil. Para esto vamos al menú de Unity y seleccionamos la opción File/Build Settings (o presiona Ctrl+Shift*B).(Mentor & Programaci, n.d.)

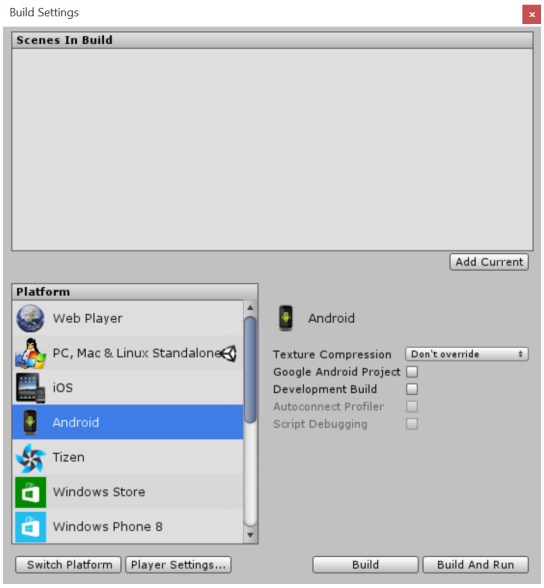

Figura 29 Apartado para la creación de la aplicación para sistemas operativos Android

Seleccionar Android e ir a la opción “Player Settings”. En la siguiente ventana se puede cambiar el Company Name y el Product Name y cargar un Icono.

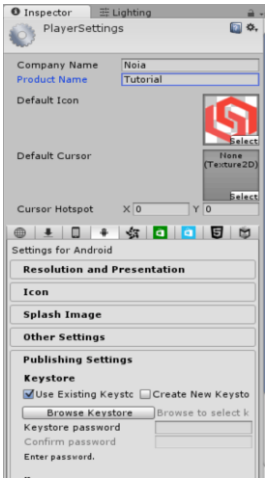

Figura 30 Configuración de características para la aplicación

Nota: Para cargar un icono basta con arrastrar una imagen de nuestro ordenador dentro del apartado “Project”, con esto el icono estará disponible dentro de Unity.

Solo queda dar click en el botón “Build” en la ventana “Build Settings”. Nos preguntará como queremos llamar a la aplicación y donde queremos guardarla. Dar Guardar y esperar a que la compile.

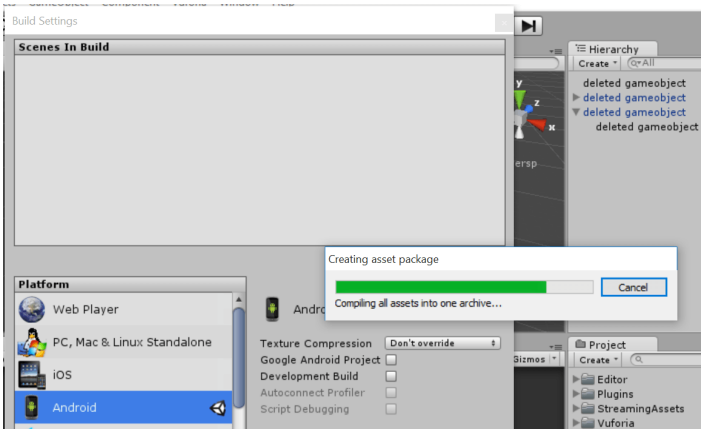

Figura 31 Creación de la aplicación

CAPÍTULO III

3. SISTEMA DE ESTIMACIÓN VISUAL-INERCIAL

En el capítulo anterior se presentó una revisión bibliográfica sobre la cual se fundamenta este proyecto de investigación. En este capítulo se muestra en detalle el proceso de implementación del sistema de odometría visual. Como se describió en el capítulo 2, el proyecto se compone de tres etapas. El primer paso y más importante es el desarrollo del sistema de estimación de visual-inercial que se explica en las siguientes secciones.

3.1.Hardware utilizado

La ejecución de la aplicación está diseñada para dispositivos que cuenten con sistemas operativo Android, a pesar de las varias versiones se opta por tener un dispositivo con la última actualización del sistema operativo considerando que el auge de la tecnología móvil avanza a grandes pasos, se busca garantizar que la aplicación este aun operativa y sea compatible en dispositivos posteriores.

La aplicación fue ejecutada en el Samsung S9 plus que cuenta con las siguientes especificaciones técnicas y de interés para el desarrollo del sistema de estimación.(“Galaxy S9 Plus: características. Galaxy S9+: precio. El mejor celular Samsung sin stylus - CNET en Español,” n.d.)(“Specifications | Samsung Galaxy S9 and S9+ – The Official Samsung Galaxy Site,” n.d.)

Especificaciones técnicas:

- Resolución: 2960 x 1440 pixeles
- Procesador: Snapdragon 845 de ocho núcleos (cuatro de 2.8GHz y cuatro de 1.7)
- RAM: 6GB
- Bateria:3500 mAh

- Sistema operativo: Android Oreo (Android 8.0.0)
- Conectividad: Wi-Fi 802.11ac(2.4 y 5GHz)
- Cámara trasera: Doble de 12 megapixeles con estabilización de imagen óptica en las dos. La principal (gran angular) tiene una apertura variable de f/1.5 a f/2.4, mientras que la secundaria (telefoto) una apertura de f/2.4
- Cámara frontal: 8 megapixeles con apertura f/1.7
- Resistente al agua: IP68 (hasta 1.5 metros de profundidad por hasta 30 minutos)
- Tamaño: 158.1x73.8x8.5mm
- Peso: 189 gramos

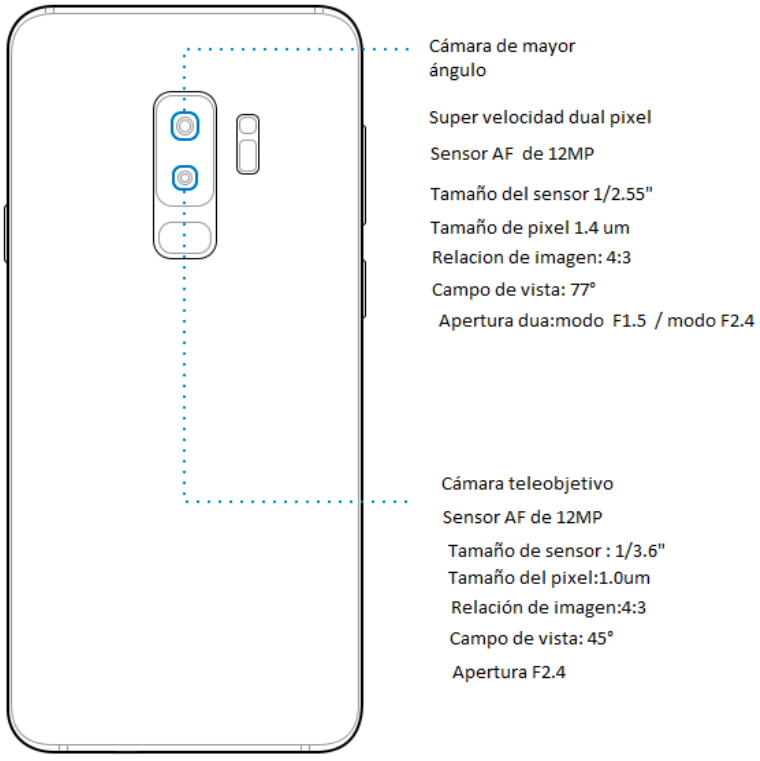

Figura 32 Características técnicas de las cámaras de Samsung S9 plus

Fuente: Samsung S9 plus (“Specifications | Samsung Galaxy S9 and S9+ – The Official Samsung Galaxy Site,” n.d.)

Procesador Snapdragon 845

Tabla 3.
Características generales del Snapdragon 845

General	
Arquitectura	64 bits
Instrucciones	ARMv8
Tecnología	10nm Samsung

Fuente: Especificaciones Samsung S9 plus ("Specifications | Samsung Galaxy S9 and S9+ – The Official Samsung Galaxy Site," n.d.)

Tabla 4.
Características de la CPU del Snapdragon 845.

CPU	
Núcleos totales	8
Principales	4x Kryo 385 a 2.8 GHz
Secundarios	4x Kryo 385 a 1.8 GHz
Ifaz. memoria	LPDDR4x a 1866 MHz
Lectura NAND	SD 3.0 (UHS-I)
Cache L1	16 KB x8
Cache L1 Datos	16 KB x8
Cache L2	1536 KB

Fuente: Especificaciones Samsung S9 plus ("Specifications | Samsung Galaxy S9 and S9+ – The Official Samsung Galaxy Site," n.d.)

Tabla 5.
Características de la GPU del Snapdragon 845.

GPU	
GPU	Adreno 630
Velocidad GPU	710 MHz
Resolución máxima	4800 x 4800 px

Fuente: Especificaciones Samsung S9 plus ("Specifications | Samsung Galaxy S9 and S9+ – The Official Samsung Galaxy Site," n.d.)

3.2.Sistema de estimación Visual-inercial

La estimación del movimiento se realiza asumiendo que la cámara está rígidamente adherida a la persona de modo que el desplazamiento y giro que se realiza sea el mismo de la cámara. La dimensión de las imágenes capturadas por el dispositivo celular son de 640 x 480. La captura de imágenes se lo realiza a una razón de 30 frames por segundo.

El algoritmo implementado para el sistema de estimación visual-inercial se desarrolló en el lenguaje de programación C# dentro del editor de la plataforma de programación Unity3D. Se utilizó la librería para visión artificial OpenCV en su versión 3.0.0 que contiene todas las funciones necesarias para la implementación de un sistema de estimación visual de movimiento.

3.2.1. Instalación de OpenCV en Unity3D

La instalación de OpenCV dentro del editor de Unity es sencilla. En del editor de Unity existe la Asset Store, allí se pueden descargar paquetes pre fabricados (escenas, objetos 3D animados, librerías, etc.) que pueden ser agregados al proyecto que se esté desarrollando. Dentro de la Asset Store se encuentra la librería de OpenCV para Unity. Para su instalación simplemente es necesario dar clic sobre el botón descargar y el editor se encarga de la instalación automáticamente, sin embargo la única desventaja es que se trata de una versión pagada. La **Figura 33** muestra la asset store de Unity donde se puede descargar la librería de visión OpenCV.

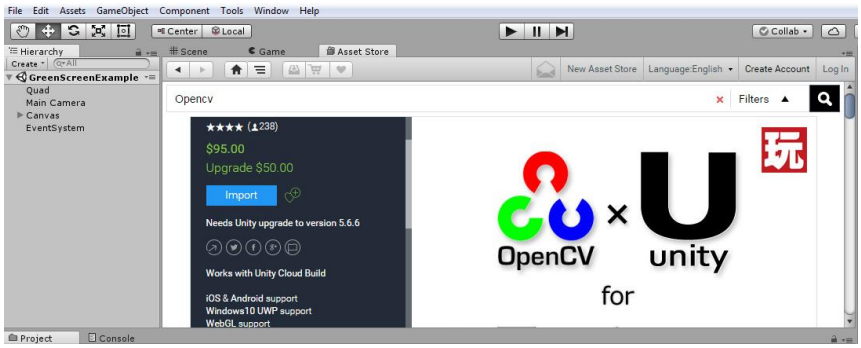

Figura 33. OpenCV para Unity

OpenCV para Unity es compatible con las plataformas Android, iOS, macOS, WebGL, UWP, Linux 32-64bits, Windows 32-64 bits. El primer paso es dar clic sobre el botón importar, el editor que desplegará una ventana con la información de los archivos que se instalarán en el proyecto que se está desarrollando. La Figura 34 muestra la carpeta plugins donde se encuentran las librerías compiladas en los lenguajes nativos para los cuales OpenCV es compatible. Adicionalmente se muestra todo el contenido a ser instalado, para ello es necesario dar clic sobre el botón importar para agregar todo el contenido al proyecto.

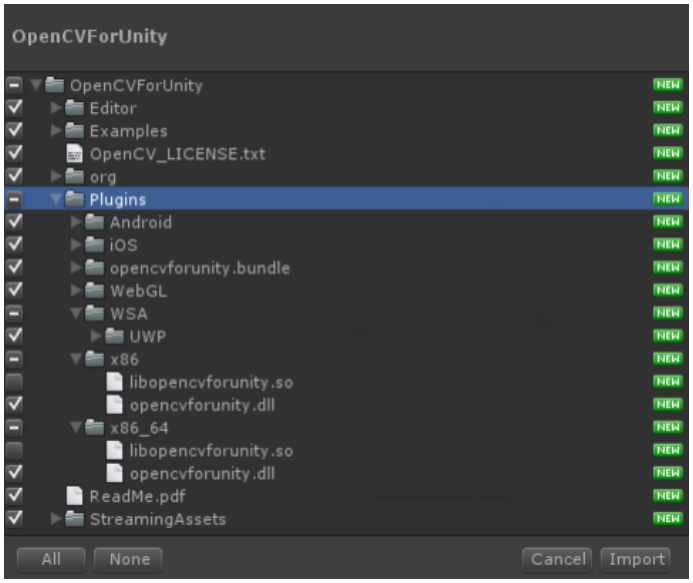

Figura 34. Paquetes de OpenCV

Fuente: OpenCV for Unity 2018

Luego de importar todo el contenido, el paso final es instalar propiamente la librería de visión al proyecto, para ello en la barra de menú del editor de Unity, accedemos a Tools\OpenCVforUnity\ Set Plugin Import Settring. Como se muestra en la **Figura 35**.

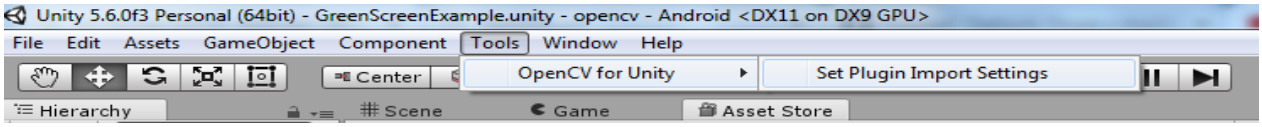

Figura 35. Instalación de la librería OpneCV

3.2.2. Implementación del sistema Visual-Inercial

En esta sección del capítulo se explica en detalle la implementación del sistema de estimación de movimiento. El proceso inicia con la captura de las imágenes del entorno con una tasa de 30 frames por segundo (FPS) que es la cantidad estándar en los dispositivos de video. Para esto Unity cuenta con la clase *WebCamTexture.*

WebCamTexture implementa todas las funciones necesarias para comunicación con las cámaras disponibles en el dispositivo que se está ejecutando, en este caso un Smartphone. Mediante la función *GetMat()* esta clase convierte la imagen capturada por la cámara del dispositivo en un matriz de imagen tipo *Mat* que es el formato con el cual la librería de visión trabaja. A continuación, el algoritmo 2 muestra el proceso de captura de imágenes.

Algoritmo2: Captura de imágenes
1: Se solicita el nombre del dispositivo
2: Mediante escaneo se guarda el número de cámaras disponibles en el dispositivo a través de un índice que identifica cada una de las cámaras encontradas.
3: Mediante el método *Play()* , el índice que identifica la cámara a usar (delantera o trasera), se enciende la cámara del dispositivo.
4: Las imágenes capturadas son almacenadas en variables tipo textura 2D, para ello se requiere: nombre del dispositivo, el ancho, alto de la imagen y los FPS de captura.
5: El método *GetMat()* convierte las texturas 2D en matrices de imágenes tipo Mat
6: Los pasos 4 y5 se repite hasta que el comando *Stop()* de apagado de la cámara sea ejecutado por parte del usuario.

Todo el sistema de estimación visual inercial está implementado en el mismo script donde se implementa la comunicación con el servidor que se explica en el capítulo 4. En esta sección se trata únicamente del sistema de estimación de pose de la persona.

Como se explica en la revisión bibliográfica del capítulo II, la estimación de movimiento visual consiste en calcular la matriz de transformación entre dos imágenes capturadas en dos instantes de tiempo consecutivos que denominaremos como t_k y t_{k-1} y la matriz de transformación la denotaremos como T_k, los subíndices k corresponden al instante de tiempo actual. El algoritmo 3 explica el proceso de cálculo de la matriz de transformación.

Algoritmo3: Cálculo de la matriz de transformación
1: Se captura la primera imagen $\boldsymbol{I_{k-1}}$
2: Se detectan puntos de alto contraste mediante un detector de características y se guardan las coordenadas de los puntos en una matriz de puntos $\boldsymbol{P}$.
3: Se captura la segunda imagen $\boldsymbol{I_k}$
4: Se detectan puntos de alto contraste mediante un detector de características y se guardan las coordenadas de los puntos en una matriz de puntos $\boldsymbol{C}$.
5: Se calcula el flujo óptico entre las imágenes $\boldsymbol{I_{k-1}}$, $\boldsymbol{I_k}$.
6: Las correspondencias entre los pares de puntos $\boldsymbol{P_i}$ y $\boldsymbol{C_i}$ se almacenan en una matriz de estado $\boldsymbol{st}$.
7: Mediante los set de puntos $\boldsymbol{C}$,$\boldsymbol{P}$ y la matriz de correspondencias, se calcula la matriz esencial $\boldsymbol{E}$
8: Se recupera la pose, rotación $\boldsymbol{R}$ y traslación $\boldsymbol{t}$ que forman la matriz de transformación $\boldsymbol{T_k}$.
9: Se copia la imagen $\boldsymbol{I_k}$ a la imagen $\boldsymbol{I_{k-1}}$ y los puntos $\boldsymbol{C}$ en la matriz de puntos $\boldsymbol{P}$ y se repiten los pasos 3-9.

Para el desarrollo del proyecto de investigación se compararon cuatro detectores de características como son SIFT, FAST, Harris, Shi-Tomasi. Los resultados se muestran en el capítulo V. A continuación, se desarrolla en detalle el proceso de detección de características en cada uno de los métodos utilizados.

3.2.2.1.Detectores de características

- **Detector SIFT**

SIFT (*Scale-invariant feature transform*) o transformación de características en escala invariante, es un detector usado en visión artificial o visión por computadora que en esencia lo que busca es extraer características distintivas de la imagen en escala de grises. Este algoritmo encuentra coincidencias entre pares de puntos comparando valores de los pixeles pertenecientes a diferentes fotografías. Mediante esto es posible reconocer una imagen dentro de una base de datos incluso dentro de otra imagen.(Flores & Braun, 2011)

Detección de extremos en el espacio-escala

En el primer paso la base es encontrar puntos invariantes a la traslación, escala y rotación de la imagen, estos puntos también deben ser mínimamente afectados por el ruido o distorsiones, Para ello se crea una pirámide de imágenes, conocidas como octavas o espacio de escala, las cuales se derivan de la imagen principal pero difieren en escala y se aplica un desenfoque en cada imagen de la octava.(Villca, 2017) Estos puntos a detectar también son considerados como puntos máximos y mínimos obtenidos a partir de la diferencia Gaussiana (DOG) (x, y, σ) , entre dos imágenes consecutivas de la misma octava, que es definida con la expresión:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (58)$$

Donde:

$$L(x, y, k\sigma) = I(x, y) - G(x, y, \sigma)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$I(x, y)$= imagen

σ = desviación estándar del filtro gaussiano conocido como factor de escala.

La Figura 36 muestra gráficamente el proceso para encontrar puntos máximos y mínimos a través de las DOG.

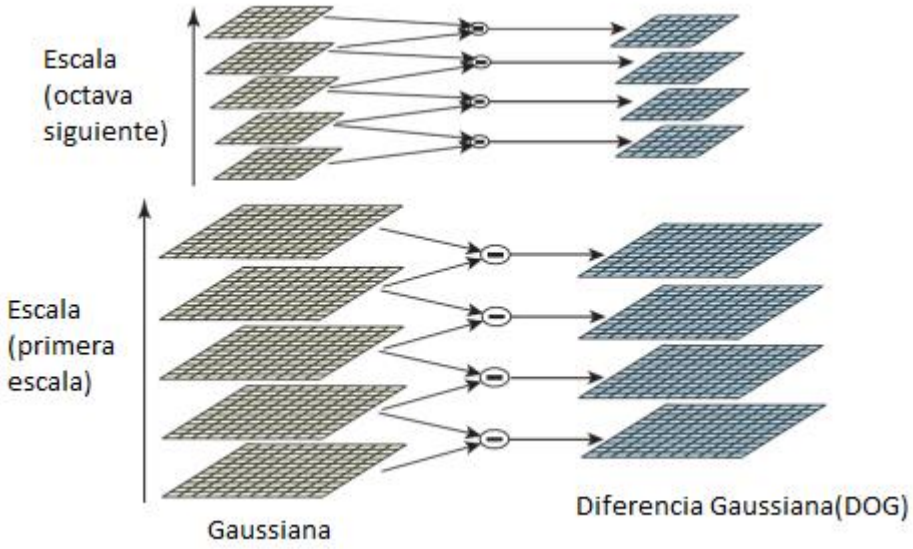

Figura 36. Diferencia Gaussiana (DOG).

Fuente: Fundamento teórico (Flores & Braun, 2011)

Localización de máximos y mínimos en imágenes DOG

Los máximos y mínimos en las imágenes que resultan de una diferencia gaussiana son los puntos característicos que se pueden encontrar en una imagen. La técnica consiste en recorrer el vecindario

de los puntos aledaños al punto de alto contraste o conocido como punto clave detectado luego de la diferencia gaussiana. Si este valor es un máximo o mínimo entonces se lo conserva como punto característico. (Villca, 2017)

Idealmente se espera que este valor máximo o mínimo coincida con un pixel, sin embargo en la realidad estos valores se encuentra en puntos medios. Computacionalmente no se puede acceder a estos valores y para ello matemáticamente se puede generar valores de subpixeles que permiten tener acceso a esta información.

Se hace uso de la herramienta matemática, las series de Taylor para hacer una estimación de la diferencia Gaussiana entorno de un punto (x_0, y_0, σ_0) , la serie de Taylor a usar será de grado 2.(Villca, 2017)

$$D(\chi) = D + \frac{\partial D^T}{\partial \chi}\chi + \frac{1}{2}\,\chi^T \frac{\partial^2 D^T}{\partial \chi^2}\chi \qquad (59)$$

Donde D y sus derivadas son evaluadas en el punto $\chi = (x, y, \sigma)^T$

Operando tenemos otra forma de expresar la ecuación:

$$D(\bar{\chi}) = D + \frac{1}{2}\,\frac{\partial D^T}{\partial \chi}\bar{\chi} \qquad (60)$$

Para eliminar puntos de bajo contraste, la técnica consiste en rechazar valores cuya magnitud de intensidad de píxel es menor a un valor umbral. Entonces tenemos lo siguiente si $|D(\bar{\chi})| < 0{,}03$ el punto es eliminado del conjunto de puntos de interés, se asume que D toma valores oscilantes entre 0 y 1.(Flores & Braun, 2011)

Luego de quitar puntos con poco contraste se debe quitar puntos que provienen de un borde puesto que son de interés únicamente las esquinas. Para ello se utiliza la matriz Hessiana H.

Si decimos que H es la matriz Hessiana de $D(x, y, \sigma)$ evaluada en un punto (x_0, y_0, σ_0) y los valores $\alpha\ y\beta$ son uno grande y otro pequeño estaremos tratando con una línea y es equivalente a:

$$Traza\ (H) = \frac{\partial^2 D}{\partial x^2} + \frac{\partial^2 D}{\partial y^2} = \alpha + \beta \quad (61)$$

$$Det\ (H) = \frac{\partial^2 D}{\partial x^2} X\ \frac{\partial^2 D}{\partial y^2} = \alpha.\beta \quad (62)$$

Si $\alpha = r.\beta$ la expresión se reescribe de la siguiente forma:

$$\frac{Traza\ (H)^2}{Det\ (H)} < \frac{(r+1)^2}{r} \quad (63)$$

Se han realizado varias pruebas para llegar a la conclusión que un umbral r=10, aumenta la relación del cuadrado de la matriz Hessiana y su determinante.

Asignación de orientación

Mediante la orientación de los puntos de la imagen los puntos de interés pueden ser descritos relativamente a estas orientaciones y de esta manera logran ser buenos candidatos para ser características invariantes a las rotaciones. Para cada punto es posible encontrar su módulo gradiente $m\ (x, y)$ y la fase $\theta\ (x, y)$

$$m\ (x, y) = \sqrt{(\Delta L_x)^2 + (\Delta L_y)^2} \quad (64)$$

$$\theta\ (x, y) = \tan^{-1}(\frac{\Delta L_y}{\Delta L_x}) \quad (65)$$

Para asegurar la invarianza a la rotación se calculan las direcciones y magnitudes de los gradientes de los píxeles cercanos al punto clave, luego se determina la orientación más grande y se la asigna al punto clave.(Vedaldi, 2007) Para esto se utiliza un histograma en el que se dividen los 360 grados de las posibles orientaciones en 36 contenedores, cada uno de diez grados. Por ejemplo si la orientación de un punto es de 25, el punto ingresa al contenedor de 20 -29 grados.(Rister, Wang, Wu, & Cavallaro, 2013)

Descriptor de puntos de interés

Luego de obtener puntos invariantes en escala y rotación, el paso final es crear una descripción única para cada punto de interés. El método consiste en crear una ventana de 16x16 entorno al punto clave, luego se divide en ventana de 4x4 como muestra la **Figura 37**. En cada ventana de 4x4 se calcula la magnitud y orientación y se coloca en un histograma de 8 contenedores, separados 45 grados uno de otro. Mediante una función gaussiana ponderada, se asignan valores a las magnitudes y orientaciones calculadas, de modo que aquellos pixeles que se encuentras más alejados del punto clave reciben menor valor que aquellos próximos al mismo. (Vedaldi, 2007)

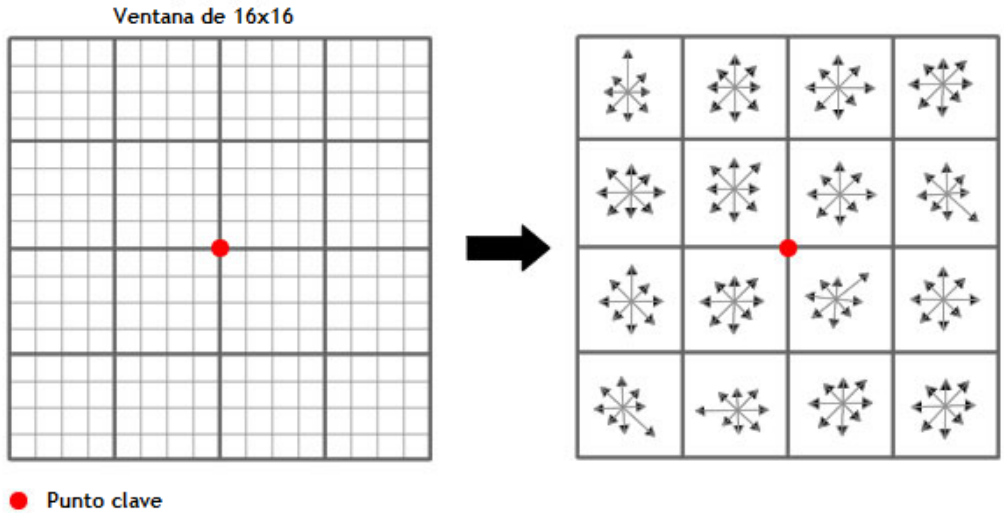

Figura 37. Descripción de puntos característicos.

Fuente: IA Shack-SIFT (2010)

Implementación

La etapa de visión está implementada en el cliente es decir será una aplicación Android y se ejecuta a través de la librería de OpenCV para Unity , si bien es cierto los algoritmos ya llevan compactada toda la matemática vista anteriormente para el método SIFT, o sin embargo hay que conocer los parámetros que se pueden modificar y como se puede ir variando los valores con el fin de obtener un algoritmo lo más óptimo posible ya que recordemos que este está siendo ejecutado en un Smartphone y conlleva cierto grado de limitaciones. El algoritmo 4 muestra la implementación del detector, incluyendo el proceso de captura de las imágenes.

Algoritmo 4: SIFT

```
1: Parámetros del detector:
puntos=100,octava=5,umbral1=0.04,umbral2=10,sigma=1.6;
2:Prev=conjuntoPuntos1,Curr=conjuntoPuntos2, KeyPoints1=conjunto1,
KeyPoints2=conjunto2;frame_gray=imagen gris
3:  sift = SIFT.create(puntos,octava,umbral1,umbral2,sigma);
4:  If Prev = 0  then
5: sift.detect(frame_gray, KeyPoints1);
6: Prev= Safe
7:    else (sift.detect(frame_gray, KeyPoints2);
8: Safe=Prev,Curr=Safe
```

- **Detector de esquinas de Harris**

Este detector tiene la ventaja de ser independiente de la escala y rotación, este detector es muy utilizado debido a su rapidez y ligereza de cómputo. Utiliza ventanas, que son pequeñas regiones de una imagen, para determinar si una ventana es idónea para contener una equina. El detector calcula una variación, entre dos imágenes consecutivas, de la ventana que se está analizando. Si la ventana produce una gran variación, entonces esta ventana contiene una esquina ("Harris Corner Detection — OpenCV 3.0.0-dev documentation," n.d.). Esta variación matemáticamente se define:

$$E(u,v) = \sum_{x,y} W(x,y)[I(x+u, y+v) - I(x,y)]^2 \quad (66)$$

Donde E es la diferencia entre la ventana original y la ventana movida ("Fundamentals of Features and Corners: Harris Corner Detector - AI Shack," n.d.) , u y v son las velocidades en la dirección x y y respcetivamente. $W(x,y)$ Es la ventana en la posición (x,y). I es la intensidad de la imagen en el punto (x,y). Usando una expansión de Taylor:

$$E(u,v) \approx \sum_{x,y} \left[I(x,y) + uI_x + vI_y - I(x,y)\right]^2 \quad (67)$$

Donde I_x, I_y son las derivadas parciales de I .Resolviendo el término cuadrático:

$$E(u,v) \approx \sum_{x,y} u^2 {I_x}^2 + 2uvI_xI_v + v^2 I_y^2 \quad (68)$$

De forma matricial:

$$E(u,v) \approx [u \quad v]\left(\sum_{x,y} W(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix} \quad (69)$$

Para simplificar la nomenclatura asignamos a una matriz M el valor de la sumatoria

$$M = \sum_{x,y} W(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (70)$$

La matriz M es utilizada por el detector para calcular la puntuación de la ventana y determinar si contiene o no una esquina. La función de puntuación viene definida de la siguiente forma:

$$R = \det M - k(trace\ M)^2 \quad (71)$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\ M = \lambda_1 + \lambda_2$$

Siendo $\lambda_1 \lambda_2$ los valores propios de la matriz M. Para ser considerados como esquinas R debe superar un valor umbral conocido como nivel de calidad.

OpenCV implementa el detector de Harris dentro de la función *GoodFeaturesToTrack*, en ella se debe especificar el número máximos de esquinas a detectar. Un punto interesante de esta función es que permite seleccionar los puntos óptimos. Por ejemplo, si en la imagen se detectan 200 puntos característicos y el programador definió apenas 100 la función toma las 100 características óptimas. Para ello la función también requiere que se defina el valor del nivel de calidad.

El nivel de calidad permite seleccionar las mejores esquinas de la siguiente forma: Si la puntuación de calidad R de una esquina es de 1000 y el nivel de calidad es 0.0, entonces todas las esquinas detectadas con una puntuación de calidad menor a 100 son rechazadas por la función. El algoritmo 5 muestra el código de programación que implementa esta función, en ella el parámetro *true* permite seleccionar el detector a utilizar. También se incluye el proceso de captura de imágenes.

- True = detector de Harris.
- False = detector de Shi-Thomasi

Algoritmo5: Harris
1: Parámetros del descriptor: puntos=100, nivel de calidad= 0.01, distancia mínima= 0.5, tamaño de la ventana=3;
2: Se captura la imagen (frame)y se convierte a escala de grises (frame_gray): cvtColor(frame, frame_gray, Imgproc.COLOR_BGR2GRAY);
3: Se detectan los puntos característicos en la imagen, se llama a la función y se definen los parámetros descritos en el primer paso : goodFeaturesToTrack(frame_gray, Corners, 50, 0.01, 0.5, mask3, 3, true, 0.04);
4: Los puntos característicos se almacenan en una matriz de puntos (Corners)
5: Se grafican los puntos detectados para observar el funcionamiento de la función : Imgproc.circle(mask, P, 7, verde); mask es la máscara sobre la que se grafica el punto P, 7 es el radio del punto y Verde es el color en código RGB (0,255,0)
6: Se repite el proceso desde el paso 2.

- **Detector de esquinas de Shi-Tomasi**

Este detector de esquinas es una variación del detector de esquinas de Harris. El cambio reside en el criterio de puntuación que permite la selección de las esquinas. A diferencia de Harris que utiliza la traza y el determinante de la matriz M. Este método de selección utiliza los valores propios de M para determinar la validez de los píxeles aceptados como esquinas.

$$R = \min(\lambda_1, \lambda_2) \tag{72}$$

Este criterio permite acelerar computacionalmente el proceso de detección de características. La Figura 11 del capítulo 2 muestra gráficamente como es el criterio de selección, en ella la región azul representa todos los valores para los cuales el pixel es considerado como una esquina. Como se mencionó en la sección anterior la función *GoodFeaturesToTrack* de la librería OpenCV implementa el detector de Shi-Tomasi. El algoritmo 6 muestra el código para la implementación de este método.

Algoritmo 6: Shi-Tomasi
1: Parámetros del descriptor: puntos=100, , nivel de calidad= 0.01, distancia mínima= 0.5, tamaño de la ventana=3;
2: Se captura la imagen (frame)y se convierte a escala de grises (frame_gray): cvtColor(frame, frame_gray, Imgproc.COLOR_BGR2GRAY);
3: Se detectan los puntos característicos en la imagen, se llama a la función y se definen los parámetros descritos en el primer paso : goodFeaturesToTrack(frame_gray, Corners, 50, 0.01, 0.5, mask3, 3, false, 0.04);
4: Los puntos característicos se almacenan en una matriz de puntos (Corners)
5: Se grafican los puntos detectados para observar el funcionamiento de la función : Imgproc.circle(mask, P, 7, rojo); mask es la máscara sobre la que se grafica el punto P, 7 es el radio del punto y rojo es el color en código RGB (255,0,0)
6: Se repite el proceso desde el paso 2.

- **Detector FAST**

Features from Accelerated Segment Test es un descriptor de esquinas considerado de alta velocidad ("FAST Algorithm for Corner Detection — OpenCV 3.0.0-dev documentation," n.d.). Este algoritmo ha sido utilizado en aplicaciones donde la detección de características en tiempo real es el principal objetivo.(Rister et al., 2013) Sin embargo cuando la capacidad de procesamiento es limitada muchos detectores potentes como SIFT no pueden ser implementados. Es por esta razón que en el 2006 Rosten y Drummond proponen un algoritmo acelerado para la detección de características FAST.

El método consiste en seleccionar un pixel p identificado como de interés con una intensidad I_p. Se define un umbral t de selección que permite discriminar si el punto p es considerado una esquina. Para ello en un vecindario de n pixeles contiguos al pixel p. El pixel p será una equina si los n pixeles cercanos son todos más brillantes en $I_p + t$ o más oscuros en

$I_p - t$ (“FAST Algorithm for Corner Detection — OpenCV 3.0.0-dev documentation,” n.d.). La Figura 38 muestra el proceso de discriminación del punto. La aceleración del método consiste en analizar únicamente cuatro de los n pixeles cercanos. En el ejemplo de la imagen primero se examinan los pixeles 1-9 y luego 5-13. Si al menos tres de estos pixeles no cumplen con las condiciones de intensidad entonces el pixel p es rechazado.

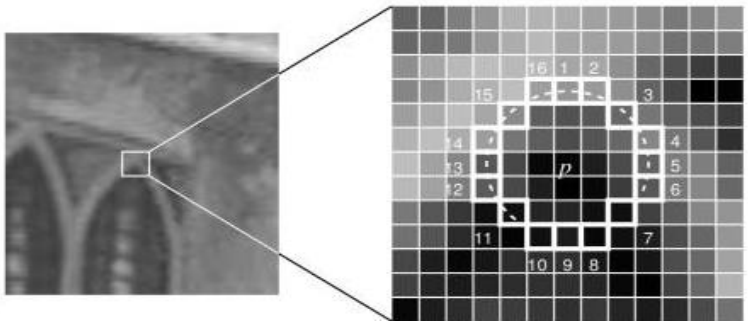

Figura 38. Detector FAST.

El algoritmo 7 muestra el proceso de implementación del detector FAST.

Algoritmo 7: FAST
1: Parámetros del descriptor: umbral=150, , máxima supresión = true; FastFeatureDetector.create(150, true, 2);
2: Se captura la imagen (frame)y se convierte a escala de grises (frame_gray): cvtColor(frame, frame_gray, Imgproc.COLOR_BGR2GRAY);
3: Se detectan los puntos característicos en la imagen, se llama a la función: fast.detect(frame_gray, KeyPoints1, new Mat());
4: Los puntos característicos se almacenan en una matriz de puntos clave(KeyPoints1)
5: Se grafican los puntos detectados para observar el funcionamiento de la función : Imgproc.circle(mask, P, 7, rojo); mask es la máscara sobre la que se grafica el punto P, 7 es el radio del punto y rojo es el color en código RGB (255,0,0)
6: Se repite el proceso desde el paso 2 para una nueva imagen

3.2.2.2.Cálculo del flujo óptico

En la sección anterior se describieron los detectores de características utilizados en este proyecto. Luego de detectar los puntos característicos el siguiente paso es el cálculo del flujo óptico entre dos imágenes consecutivas con el fin de encontrar las correspondencias de puntos y poder estimar posteriormente la matriz de transformación que permita determinar el movimiento de la persona. Para ello OpenCV implementa una función que permite resolver las ecuaciones del flujo óptico, desarrolladas en el capítulo 2.

OpenCV implementa el solucionador de Lukas-Kanade (capítulo 2) para hallar los vectores de flujo óptico que describen el movimiento de un vecindario donde una esquina ha sido detectada. Luego estos vectores son comparados con los vectores correspondientes al vecindario de la esquina que se encuentra en la siguiente imagen capturada para determinar la correspondencia de los pixeles. Estas correspondencias se almacenan en una matriz de estado, formando parejas de esquinas o puntos con sus respectivas coordenadas en píxeles.

La función que implementa este solucionador en OpenCV es *calcOpticalFlowPyrLK().* Para ello el método requiere de los puntos o esquinas encontradas en la imagen tomada en el tiempo $k-1$, las esquinas encontradas en la imagen actual es decir en el tiempo k. Esta función retorna la matriz de estado st en la cual almacena las correspondencias y coloca una bandera en 1, si la característica encontrada en la imagen I_{k-1} ha sido encontrada en la imagen I_k. Caso contrario la bandera es colocada en 0. La Figura 39 muestra la línea de código necesario para implementar el flujo óptico.

```
Video.calcOpticalFlowPyrLK(old_gray, frame_gray, Prev, Curr, st, err, WindowSize, 3);
```

Figura 39. Cálculo del Flujo óptico

Los parámetros $old\_gray, frame\_gray$ son las imágenes I_{k-1}, I_k respectivamente, transformadas a imágenes en escala de grises para ello la función *cvtColor()* permite la conversión de una imagen RGB a una imagen en gris y viceversa.

Hasta este punto se han capturado las imágenes en frames consecutivas, se han detectado las características en ellas y se ha calculado el flujo óptico de los puntos característicos entre las imágenes capturadas, sin embargo en el proceso de implementación se presentó un inconveniente. El proceso de respiración de las personas implica un ligero movimiento de la cámara hacia arriba y hacia abajo. Cabe mencionar que la cámara se encuentra adherida al pecho de la persona. Este movimiento causa errores en el proceso de cálculo del flujo óptico y por ende en el proceso de estimación de movimiento. Para ello se definió un umbral de movimiento, de forma experimental. Este umbral es la distancia la euclidiana que se produce por el proceso de respiración cuando la persona se está de pie sin moverse.

$$D = \sum_{n} \sqrt{\left(C_{i,x} - P_{i,x}\right)^2 + \left(C_{i,y} - P_{i,y}\right)^2} \quad (73)$$

$$dist\_umbral = \frac{D}{n} \quad (74)$$

La Figura 40 muestra la implementación de la distancia umbral que evita estimar falsos movimientos mientras la persona se encuentra estática. Esta distancia es calculada en pixeles.

```
for (i = 0; i < st.rows(); i++)
{
    Pst = new Point(st.get(i, 0));
    if (Pst.x == 1)
    {
        P = new Point(Prev.get(i, 0));
        //P2 = lp2[i];
        P2 = new Point(Curr.get(i, 0));
        double D = Mathf.Sqrt((Mathf.Pow((float)(P2.x - P.x), 2)) + (Mathf.Pow((float)(P2.y - P.y), 2)));
        Imgproc.circle(mask, P2, 7, scalar3);
        Imgproc.circle(mask, P, 7, scalar2);
        Imgproc.line(mask, P, P2, scalar3);
        d = d + D;

    }
}
double distance = d / st.rows();
```

Figura 40. Umbral de movimiento.

3.2.2.3.Cálculo de la matriz esencial y recuperación de la pose

En la sección anterior se calculó el flujo óptico de las características encontradas por los detectores. Luego de calcular el flujo óptico y encontrar las correspondencias 2D-2D de puntos entre las dos imágenes, el siguiente paso es calcular la matriz esencial E que contiene la matriz de transformación T. Recordando la formulación matemática tenemos:

$$p'_i = Rp_i + t \tag{75}$$

Donde R y t son las matrices de rotación y traslación y p'_i, p_i son los puntos característicos que cumplen con la correspondencia 2D-2D en las imágenes I_k, I_{k-1} respectivamente. Esta ecuación determina la correspondencia entre las imágenes capturadas consecutivamente y la solución de la misma determina los valores de rotación y traslación realizada por la cámara del en el intervalo de tiempo $k-1, k$. Reformulando la ecuación tenemos:

$$p'_i{}^T E\, p_i = 0 \qquad (76)$$

Donde $E = tR$, es la matriz esencial. La librería de visión OpenCV implementa la función *findEssentialMat()*. La función requiere los puntos característicos encontrados en las imágenes I_k, I_{k-1} y la matriz de correspondencia st que contiene las banderas afirmando la validez de la correspondencia entre cada par de puntos p'_i, p_i. Adicionalmente es necesario ingresar los valores intrínsecos de la cámara como la distancia focal y el punto central de imagen de la misma. Estos valores se encuentran mediante la calibración de la cámara. La Figura 41 muestra la línea de código que implementa esta función.

```
E = Calib3d.findEssentialMat(Prev, Curr, focal, pp, Calib3d.LMEDS, 0.90, 3, st);
Calib3d.recoverPose(E, Prev, Curr, R, t);
```

Figura 41. Matriz esencial y recuperación de la pose.

El siguiente paso es descomponer la matriz esencial E para hallar las matrices de Rotación y traslación que permitan recuperar la pose de la cámara. Y para esto se utiliza la función r*ecoverPose()* de OpenCV. Este método retorna las matrices R y t como muestra la Figura 41 .

Muchos sistemas de estimación utilizan una fórmula que combina la matriz rotación y traslación para trazar la trayectoria que ha realizado la persona. Sin embargo, en este proyecto se utiliza únicamente la matriz de traslación que nos permite conocer el avance realizado por la persona. Para determinar los ángulos de giro se utilizaron los datos de la IMU incorporada en el dispositivo móvil. En la siguiente sección se explica cómo se utilizaron los datos inerciales para trazar la trayectoria.

3.2.2.4.Reconstrucción de la trayectoria y fusión de datos inerciales.

Resumiendo el proceso realizado en los apartados anteriores hasta ahora se han capturado las imágenes en frames consecutivos, se han detectados los puntos característicos o esquinas en cada una de las imágenes capturadas y se ha determinado la correspondencia 2D-2D de los puntos mediante el cálculo del flujo óptico. Para posteriormente recuperar la pose mediante el cálculo de las matrices de rotación y traslación.

El paso final es reconstruir la trayectoria realizada por la persona, para ello se suman cada uno de los desplazamientos en cada instante de tiempo k determinados en la matriz de traslación t_k, para obtener la distancia total de desplazamiento.

$$t_f = t_k + t_f$$

Donde t_f es el desplazamiento que va acumulando cada desplazamiento instantáneo y permite trazar la trayectoria final. Sin embargo, con los desplazamientos la trayectoria no está completa, se debe estimar el ángulo de giro que realizó el individuo.

Los datos inerciales proporcionados por el giroscopio del dispositivo. El entorno de programación Unity cuenta con la clase *gyro*. Esta clase permite acceder a los datos de acelerómetro y giroscópio del dispositivo. Mediante el método *attitude.eulerAngles.* La función retorna los ángulos de Euler. Estos ángulos permiten conocer la orientación del dispositivo respecto a los ejes cardinales. Así por ejemplo si el teléfono está alineado al eje norte el ángulo de Euler respecto al eje Z marcará 0 grados, al girar el dispositivo hacia el oeste el ángulo marcará 90 grados , etc. En base a estos datos se desarrolló un sistema de discretización que permite cambiar las coordenadas (x, y) de la trayectoria en relación a la dirección de avance. La Figura 42 muestra el código implementado.

```
///Norte
if ((rotacion.eulerAngles.z < 56 && rotacion.eulerAngles.z > 0) || (rotacion.eulerAngles.z > 313))
{
    Pst2.y = (-t.get(1, 0)[0]) + Pst2.y;
    Pst2.x = ((-t.get(0, 0)[0]) / 3) + Pst2.x;
    Imgproc.circle(mask2, Pst2 * 0.7, 3, Verde, 3);
    angulos.x = rotacion.eulerAngles.z - 15;
}
///Oeste

if (rotacion.eulerAngles.z > 56 && rotacion.eulerAngles.z < 120)
{
    Pst2.x = (-t.get(1, 0)[0]) + Pst2.x;
    Pst2.y = ((-t.get(0, 0)[0]) / 3) + Pst2.y;
    Imgproc.circle(mask2, Pst2 * 0.7, 2, Anaranjado, 2);
    angulos.x = rotacion.eulerAngles.z - 25;
}
///Sur
if ((rotacion.eulerAngles.z > 120 && rotacion.eulerAngles.z < 197.5))
{
    Pst2.y = (t.get(1, 0)[0]) + Pst2.y;
    Pst2.x = ((-t.get(0, 0)[0]) / 3) + Pst2.x;
    Imgproc.circle(mask2, Pst2 * 0.7, 3, Verde, 3);
    angulos.x = rotacion.eulerAngles.z - 5;

}
///Este
if (rotacion.eulerAngles.z > 197.5 && rotacion.eulerAngles.z < 313)
{
    Pst2.x = (t.get(1, 0)[0]) + Pst2.x;
    Pst2.y = ((-t.get(0, 0)[0]) / 3) + Pst2.y;
    Imgproc.circle(mask2, Pst2 * 0.7, 2, Anaranjado, 2);
    angulos.x = rotacion.eulerAngles.z - 15;
}
```

Figura 42. Reconstrucción de la trayectoria.

Finalmente, estos datos de trayectoria se almacenan en el punto Pst2 que muestra la imagen anterior y son enviados al servidor para que se pueda realizar la trayectoria dentro del entorno virtual.

3.2.2.5. Cálculo de la altura

El sistema de estimación de movimiento no solo calcula la posición de la persona en un plano 2D, también puede determinar la altura o piso en el que se encuentra el agente dentro del edificio. Mediante los datos del barómetro integrado en el Smartphone, se calcula la altura respecto al nivel del mar. Esta altura se registra como el primer piso del edificio. Si se detecta un cambio de altura superior a 3 metros, se detecta como un cambio de piso. La fórmula de cálculo se presenta en el capítulo 2 y se implementó e lenguaje nativo Android, a diferencia del resto del proyecto desarrollado en C#. El código de Android se compiló como plugin para que pueda ser usado dentro del entorno de Unity. (Asociados et al., 2005)

Para crear un plugin, desde el entorno de desarrollo de Android o SDK, se crea un proyecto de tipo librería. Luego de realizar la programación necesaria en el menú Build se compila el proyecto. Una vez compilada la librería se accede a la siguiente dirección desde el explorador de archivos: AndroidStudioProjects\AndroidProject\unityplugin\build\intermediates\packaged-classes\debug. Dentro de la carpeta se copia el archivo class.jar y se importa al proyecto de Unity.

Luego de importar el plugin en Figura 43 se muestra el código que permite obtener los datos del barómetro (Mentor & Programaci, n.d.)

```
if (plugin != null)
{
    float sensorValue = plugin.Call<float>("getSensorValues", "accelerometer");

    altura = GetDeviceAltitude(sensorValue);

    acelerometroy.text = "altura:" + altura;
}
```

Figura 43.Acceso a datos del plugin.

En la Figura 44 se muestra la función que convierte los datos del barómetro en datos de altura. En ella se implementa la fórmula presentada en el capítulo 2.

```
private float GetDeviceAltitude(float pressure)
{
    if (pressure == 0)
    {
        return 0;
    }
    return 44330.0f * (1.0f - Mathf.Pow(pressure / StandardAthmosphere, AltitudeCoef));
    //  return -7 * Mathf.Log((pressure / 1000) / (pressureAtSeaLevel / 1000)) * 1000;
}
```

Figura 44. Cálculo de la altura.

Finalmente, el desarrollo de la simulación de movimiento se realiza en el capítulo 4. En el capítulo 5 se muestran los resultados de la implementación del sistema de estimación Visual-Inercial desarrollado en este capítulo.

CAPÍTULO IV

4. TRANSMISIÓN DE DATOS Y SIMULACIÓN DE MOVIMIENTO

4.1.TRANSMISIÓN DE DATOS CLIENTE-SERVIDOR

El Proyecto de estimación visual-inercial de movimiento utiliza un sistema de transmisión de datos inalámbrico a través de una red WiFi establecida entre el Smartphone y el computador. Este sistema de comunicación es de tipo cliente-servidor TCP, siendo el Smartphone el dispositivo cliente que se encarga de enviar los datos de pose (desplazamiento y giro) hacia el dispositivo servidor, un computador que funciona como una estación de monitoreo de la trayectoria de los efectivos militares.

4.1.2. Creación de la red Wifi

La comunicación es un apartado fundamental en el proceso de todo el sistema, el cliente hace la mayor parte a nivel computacional ejecutando algoritmos de visión y por otro lado el servido está encargado de proyectar la parte grafica que se traduce en la interpretación de lo previamente calculado en el *smartphone* , partiendo de este criterio es necesario enlazar estos sistemas para que trabajen de forma óptima y coordinada , el cliente es un sistema encargado de enviar la pose y datos como altitud , mientras que el receptor se encarga de interpretar dichos datos para representar gráficamente con animación de objetos en 3D y graficas en 2D.

Por facilidad de implementación se montó un red WLAN sin embargo para montar un infraestructura de gran escala se necesita de varios equipos. Para fines más prácticos se optó por usar un Virtual Wi-Fi Router, es una aplicación portable que permite convertir el ordenador en un router virtual, esto permitirá “amplificar” la señal de Wifi con servicio de internet , sin embargo

solo se hará uso del medio de transmisión. Es evidente que el alcance también es un defecto pero para el aplicativo funciona muy bien ya que se está enviando algunos caracteres en forma de *strings* en cada trama.

- **Configuración**

  Abrimos el CMD de computador como muestra la Figura 45.

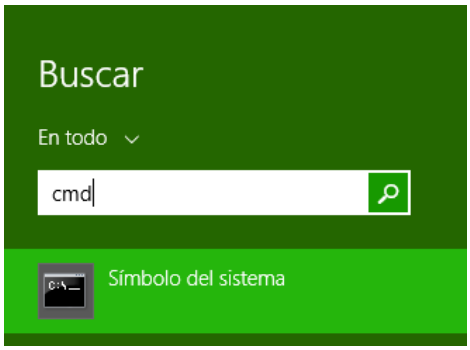

Figura 45 Búsqueda del CMD

Dentro del CMD se configura la red de forma rápida y por medio de comandos. La Figura 46 muestra la interfaz del CMD.

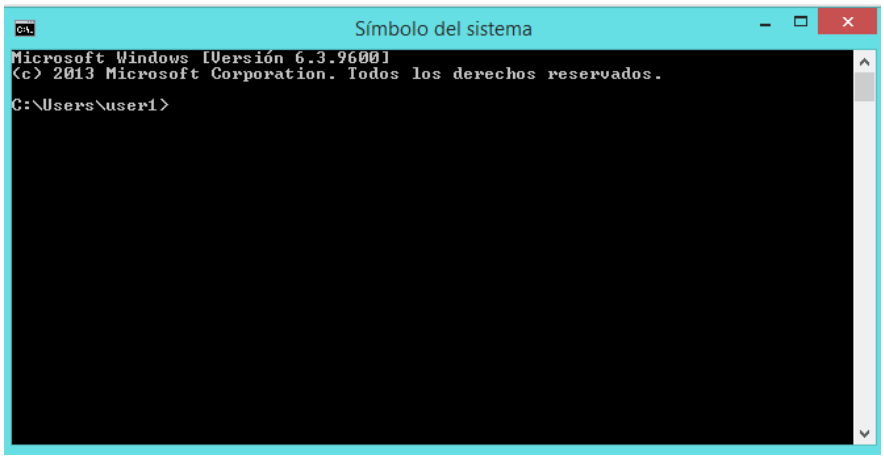

Figura 46 Interfaz del CMD

Se introduce el primer comando que describe toda la configuración de una red Wifi, se puede editar el nombre de la red y el *password* , una clave de acceso a la red de mínimo 8 caracteres. La línea de comandos se muestra a continuación y en la Figura 47 .

netsh wlan set hostednetwork mode=allow ssid=nombre_de_la_red_wifi key=Password_de_la_wifi

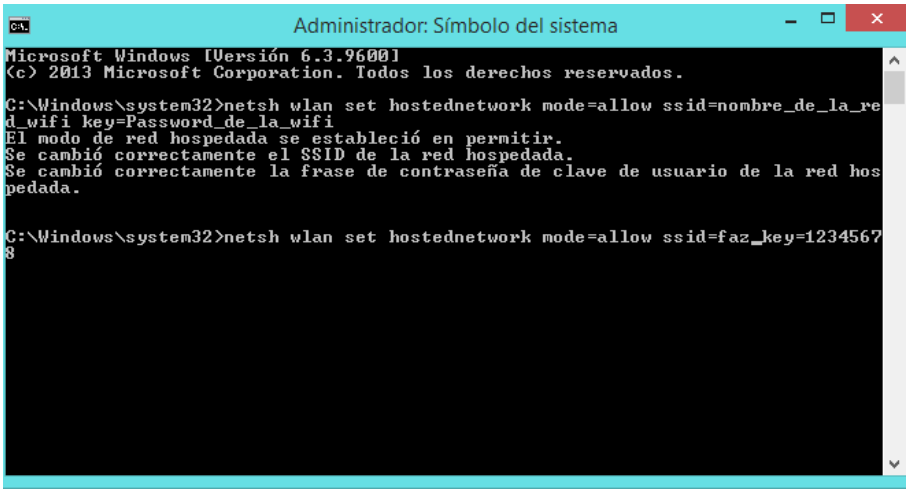

Figura 47 Configuración de la red

Una vez configurada la red se le debe inicializar mediante le siguiente comando. Como muestra la Figura 48.

netsh wlan start hostednetwork

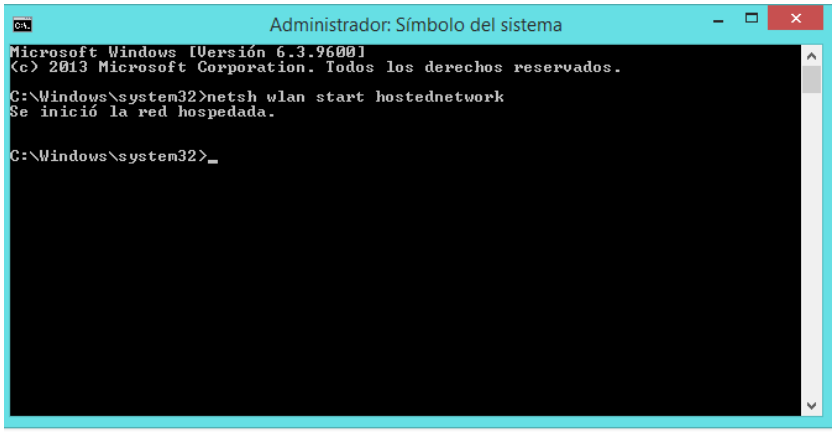

Figura 48 Inicialización de la red

Para detener la ejecución de la red se debe colocar la siguiente línea de comandos como muestra Figura 49.

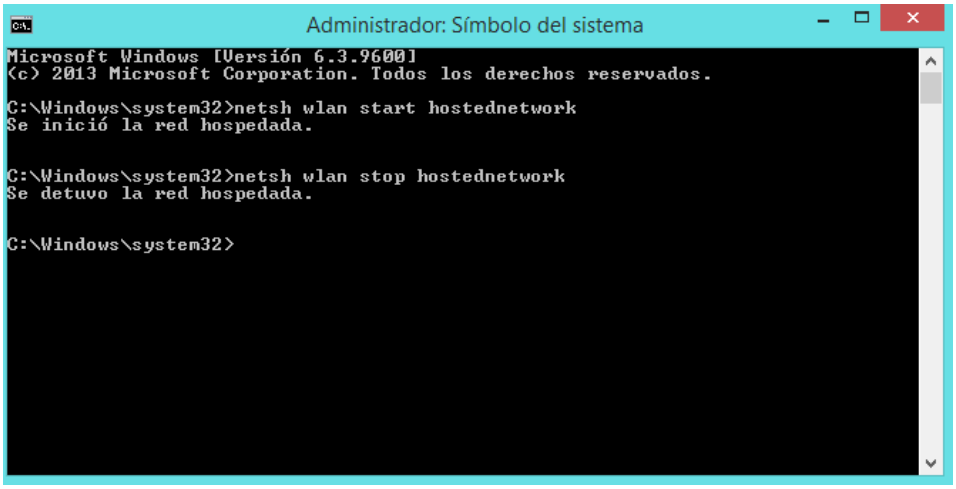

Figura 49 Detener a la red

Con los pasos anteriores establecemos la comunicación del cliente y servidor, haciendo uso del Wifi virtual del computador cabe hacer mención que no es una red de gran alcance, pero práctica para realizar las pruebas del sistema.

4.1.3. Algoritmo Cliente

El script Cliente utiliza el namespace System.Net.Socket de Microsoft, que se conoce como Winsock. El cual contiene todas las clases necesarias para implementar un sistema de comunicación. En este proyecto se utilizó la clase TcpClient para instanciar el socket utilizado en el desarrollo de esta aplicación. El socket utiliza la dirección IP del ordenador y el número de puerto. Las direcciones IP y el puerto utilizado se muestran en la Tabla 6. Mediante el método ConnectToserver() implementado en el algoritmo se crea el socket utilizando la información de la Tabla 6.

Tabla 6.
Direcciones IP y puerto.

Smartphone-cliente	**192.168.173.2**
Ordenador-Servidor	192.168.173.1
Puerto	6321

- **Método Send()**

Mediante este método el dispositivo cliente envía la información de pose hacia el servidor. Para esto se utilizó la clase NetworkStream. Esta clase utiliza dos objetos para la lectura y escritura de datos a través de la red. El objeto StreamReader lee una línea de datos que se encuentra en el *stream* actual en formato *string*. El objeto StreamWriter escribe una línea de datos en un stream en formato *string*, seguido de su respectivo terminador.

El intercambio de datos cliente-servidor, se realiza mediante datos en formato string exclusivamente, de modo que es necesaria una conversión a formato float para que puedan ser utilizados por el algoritmo de simulación del movimiento. Este tipo de comunicación representa una desventaja cuando la estructura de los datos es compleja. En el caso de este proyecto se requiere el envió de dos *arrays* con información de traslación y rotación respectivamente. Desde el cliente se envía en un mismo *string* los arrays de traslación y rotación separados por un carácter que le permite a la función discretizar() implementada en el servidor para separar los datos. Se debe tener en cuenta que este proceso de discretización de datos puede representar un tiempo de ejecución alto, computacionalmente hablando, que como resultado generará retrasos entre el movimiento de los efectivos militares y la simulación de la trayectoria. Para reducir este efecto la función discretizar() fue optimizada mediante saltos de posiciones dentro del string recibido por el servidor.

4.1.4. Algoritmo Servidor

En la Figura 50 se presenta la clase Server desarrolladas dentro de la programación del proyecto. Como se pude observar la clase Server hereda atributos de la clase ServerClient y posee varios métodos que se explican a continuación.

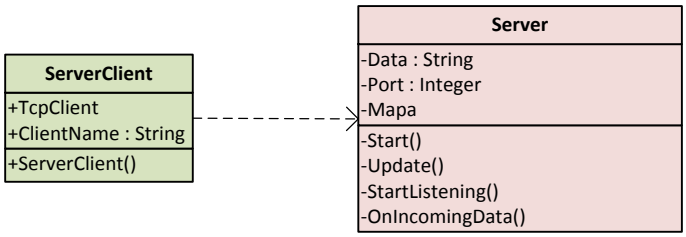

Figura 50. Diagrama UML de la clase Server.

- **Función Start()**

La transmisión de datos se inicializa en esta función mediante la clase TcpListener, para establecer la comunicación esta clase toma como atributos la dirección IP del ordenador o dispositivo sobre el que se ejecute el programa Server y adicionalmente el número del puerto por el cual se enviaran los datos.

- **Función StartListening()**

Una vez que se ha establecido la comunicación con la clase TcpListener, mediante el método StarListening, se inicia la escucha de datos a recibir.

- **Función Update()**

Luego de inicializar las variables y la comunicación, dentro de la función Update se desarolla la simulación del movimiento. A través del atributo Data se reciben los datos enviados por el

dispositivo móvil. Luego de la recepción del dato tipo string , se realiza la conversión a un valor tipo float para ser utilizado el componente Trasnform de la cámara del entorno virtual. Al modificar el componente mencionado, se crea la simulación del movimiento. El componente Transform contiene las coordenadas X,Y,Z de la cámara dentro del entorno y los grados de giro entorno a los ejes X,Y,Z

- **Funcion OnIncomingData()**

A través del método readLine, la clase NetworkStream detecta un dato dentro del canal de comunicación establecido y recibe el dato enviado desde el Cliente. Una vez recibido el dato la función OnIncomingData realiza la conversión de los datos de tipo String a tipo float y además separa el vector con datos de desplazamiento del vector con los datos de giro.

4.2.DESARROLLO DEL ENTORNO DE REALIDAD VIRTUAL

El entorno virtual desarrollado es una versión digital del edificio central de la Universidad de las Fuerzas Armadas ESPE. Donde se desarrollaron las pruebas de funcionamiento del proyecto. En esta sección, se describe el proceso de creación del entorno mencionado.

4.2.2. Digitalización del espacio

Para la digitalización del ambiente urbano, se utilizó el software de modelamiento 3D Sketch Up debido a su completo portafolio de herramientas de diseño y su librería pública de modelos en línea conocida como 3D Warehouse. El modelo base utilizado se encuentra disponible en la librería de SketchUp 2016 (“Edificio Central | 3D Warehouse,” n.d.).

Fue desarrollado en base a los planos del edificio. Cuenta con los bloques de las aulas A y B y el área administrativa. Se encuentra georreferenciada por lo que puede ser vista a través de Google

Earth (“Edificio Central | 3D Warehouse,” n.d.). La escala del modelo es de 1:1 con el fin de facilitar la simulación de la trayectoria de los efectivos militares. La Figura 51, muestra la escala del modelo.

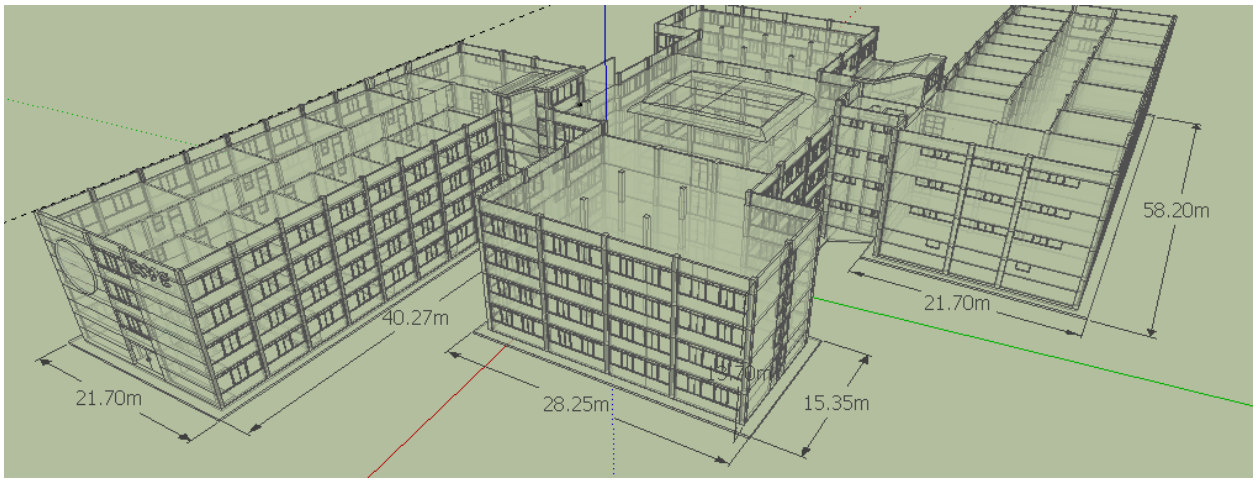

Figura 51. Escala el modelo.

El primer paso es trazar la estructura del edificio como se muestra en la Figura 51, debido a que el edificio posee cuatro pisos, la digitalización se realizó por etapas, una por cada piso como se presenta en la Figura 52 . Posterior al trazo de la estructura, se colocan las texturas del modelo con el objetivo de darle realismo al entorno virtual.

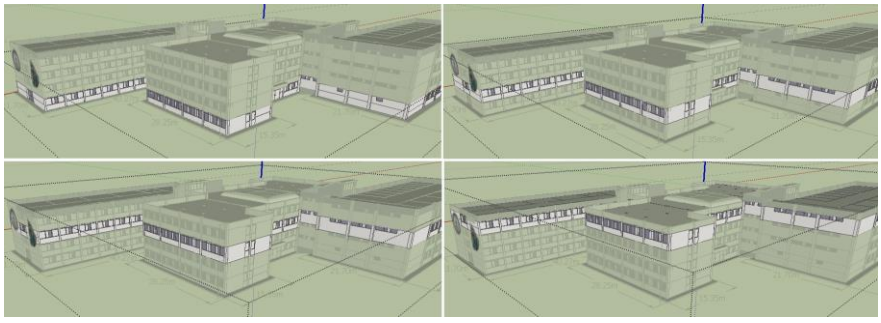

Figura 52.Etapas de digitalización

El entorno de Sketch up cuenta con la bandeja de texturas desde la cual son seleccionadas. Permite además incluir fotografías como texturas. La Figura 53, muestra la fotografía del escudo del edificio central, incluido como textura en el modelo.

Figura 53. Texturas en el modelo digitalizado.

EL paso final es exportar el modelo al entorno de desarrollo de software Unity 3D. El formato seleccionado es el formato FBX debido a que es uno de los formatos estándar para diseños 3D, permite el desarrollo de animaciones (“FBX binary file format specification — Blender Developers Blog,” n.d.). Y es soportado por el editor gráfico de Unity (“Unity - Manual: Formatos 3D,” n.d.) . La ventaja principal de este formato, es la posibilidad de exportar el modelo junto con las texturas asignadas en el software de modelamiento, hacia el editor de software Unity. En la Figura 54 se muestra el modelo finalizado.

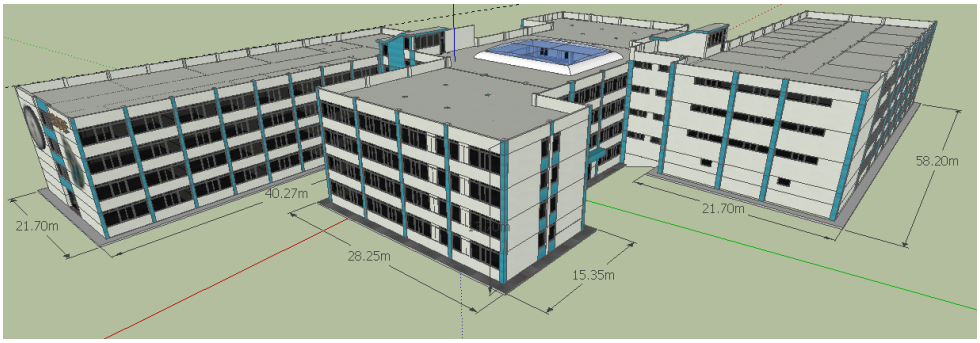

Figura 54. Modelo finalizado.

4.2.3. Preparación del entorno.

Un entorno virtual debe contar con varias características físicas para que la percepción de realidad sea mayor. Para esto Unity cuenta con una extensa lista de componentes que se agregan a los objetos de la escena (*gameObjects*) y permiten dotar de características físicas como colisiones, gravedad y otras fuerzas ("Unity - Manual: Physics," n.d.), a estos objetos. Unity cuenta con dos tipos de física o como se lo conoce como *physics-engine*. La primera orientada al desarrollo de aplicaciones en 2D y la segunda hacia aplicaciones en 3D implementada en este proyecto.

RigidBody es uno de los componentes para el control de la física, más importante dentro del desarrollo de aplicaciones en 3D. Este componente permite simular fuerzas como torque y gravedad, con el objetivo de obtener un movimiento de forma realista ("Unity - Scripting API: Rigidbody," n.d.).

En este proyecto se implementó el *gameObject FPScontroller*, que contiene la cámara del entorno y se encarga de ejecutar el desplazamiento dentro del entorno. *FPScontroller* contiene el componente *RigidBody* que permite agregar una masa al objeto de modo que se sujete al piso del entorno. Adicionalmente se agregó el componente *meshCollider* que evita que los objetos dentro de la escena se atraviesen entre sí.

Finalmente se agregaron *terrains* que son *gameObjects* que sostienen todos los objetos dentro de la escena. La Figura 55 muestra el entorno virtual finalizado.

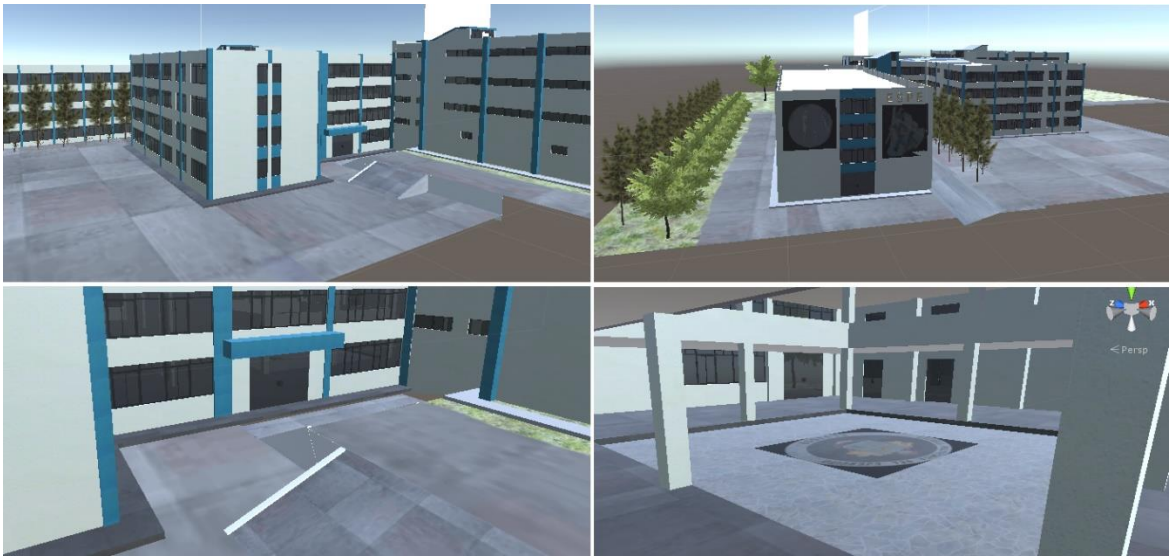

Figura 55. Entorno virtual.

4.2.4. Simulación del movimiento

Con el objetivo de monitorear la posición y orientación de los efectivos militares, se desarrolló una versión digital del entorno urbano. Las animaciones de desplazamiento y giro sobre el entorno se realizaron en la plataforma de desarrollo de software Unity 3D.

El programa de simulación desarrollado (Script) se compone de dos partes, la primera permite la recepción de los datos de estimación mediante streams por parte del dispositivo móvil a través de una comunicación TCP. La segunda parte del programa utiliza los datos recibidos y emula el movimiento del efectivo dentro del entorno virtual.

Este Script comprende el Servidor del sistema de transmisión de datos, y se ejecuta sobre un computador portátil. Las características del computador se muestran en la

Tabla 7.

Tabla 7.
Características del ordenador.

Marca	**Dell**
Sistema Operativo	Windows 8.1
RAM	16 GB
ROOM	500 GB
Procesador	Core i7
Tarjeta de video	NVIDIA Physx
Dirección IP	192.168.173.1
Puerto utilizado	6321

Para realizar el giro se utiliza el método de los cuaterniones, utilizado en robótica para representar de una forma sencilla giros en un espacio tridimensional (“Unity - Scripting API: Quaternion,” n.d.). Para emular el avance de la persona se utilizan los datos de desplazamientos obtenidos por los algoritmos de visión. Puntual mente se utiliza la matriz t (capítulo 2) que nos indica el desplazamiento estimado entre dos frames consecutivos. Estos datos recibidos por el sistema de conexión Wi-Fi permiten modificar el componente *transform*. *Transform* permite controlar la posición y rotación del objeto al que se encuentra agregada, como se puede observar en la Figura 56.

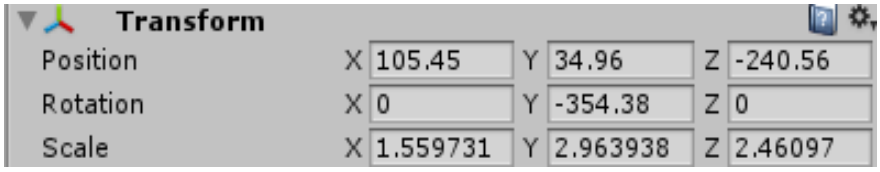

Figura 56. Componente *Transform* para la simulación del movimiento.

CAPÍTULO V

5. PRUEBAS Y RESULTADOS

En los capítulos anteriores se explicó el proceso de implementación del sistema de estimación visual-inercial. En este capítulo se presentan los resultados obtenidos durante diferentes pruebas realizas. Estas pruebas se realizaron en el bloque central del campus de la Universidad de las Fuerzas Armas ESPE. El dispositivo móvil se encuentra adherido a un soporte que sostiene al dispositivo a la altura del pecho. Mediante los ángulos de Euler obtenidos del giroscopio, se pudo definir un ángulo de inclinación que permite tener una escala fija en las imágenes captadas por la cámara.

En las siguientes secciones del capítulo se explica en detalle el proceso de evaluación del sistema y se presenta de forma gráfica los resultados de los mismos.

5.1.Métricas de evaluación

Los sistemas de odometría visual están ligados a las condiciones del entorno sobre el que se desarrolla el proceso de estimación de movimiento. Estas condiciones pueden ser iluminación, fondo estático o en movimiento, características de la textura del entorno. Es por eso que se han ido desarrollado varios métodos para el procesamiento de imágenes, especialmente en la detección y descripción de puntos característicos. Para evaluar el desempeño del sistema implementado en este proyecto de investigación, se utilizaron 4 métodos de detección de características.

Los detectores de características o puntos de interés están clasificados en dos grupos, de esquinas y *blobs* (texturas, color). Se planteó el uso de cuatro, SIFT, FAST, SHI-TOMASI Y HARRIS es decir se puso a prueba dos de cada grupo, siendo detectores de esquinas Shi-Tomas y Harris, y detectores de blobs SIFT y FAST. Cabe hacer mención de FAST es como un detector intermedio de ambos grupos.

Analizando los parámetros de desempeño de los detectores se consideró que la iluminación es un factor que afecta directamente al sistema. Dependiendo de la hora y del nivel de iluminación los detectores actúan de distinta forma por lo que se planteó medir la iluminación en tres horarios distintos para realizar las pruebas. La medida de iluminación vendrá proporcionada por el luxómetro incorporado en el dispositivo Smartphone.

Para finalizar las pruebas se vio conveniente conocer el error de desplazamiento entre la trayectoria real realizada y la estimada sobre el plano 2D y el ambiente virtual. Esta es la prueba más importante y de mayor interés ya que se debe evidenciar la exactitud del sistema de estimación. Se realizaron 3 trayectorias diferentes donde están inmersas giros y los desplazamientos posibles a realizar dentro del entorno. Las trayectorias realizadas permiten medir el error acumulativo que puede presentarse en este tipo de sistemas es por eso que en cada trayectoria se aumentó la distancia recorrida.

Los errores que permiten evaluar el sistema se calcularon con las siguientes fórmulas:

Error relativo

$$\varepsilon_R = \frac{Valor_{medido} - Valor_{teórico}}{Valor_{teórico}} \qquad (77)$$

Error relativo Porcentual

$$\varepsilon_R\% = \varepsilon_R * 100\% \quad (78)$$

EMC (Error cuadrático medio)

$$ECM = \frac{1}{n} * \sum_{i=1}^{n} (\widehat{y_i} - y_i)^2 \quad (79)$$

Donde n= número de datos

$\widehat{y_i}$ = vector de valores estimados

y_i=vector de valores reales

5.2.Resultados.

Los resultados de las pruebas realizadas se muestran de forma gráfica mediante la trayectoria realizada. Sobre el plano del edificio se presenta la trayectoria real (verde) y la trayectoria estimada (roja). La clasificación de las pruebas se realizó mediante tres trayectorias, al final de cada una se presenta un cuadro de errores entre el desplazamiento realizado y el estimado por el sistema.

- **Trayectoria 1**

Esta prueba se realizó a las 10 am con un nivel de iluminación promedio de 576,66 lux.

Detector SIFT

La trayectoria resultante en este método dista mucho de la trayectoria realizada, con un error porcentual de 67% y un error cuadrático medio de 163,36. Se decidió omitir las pruebas con este detector debido a la carga de procesamiento requerido para su ejecución.

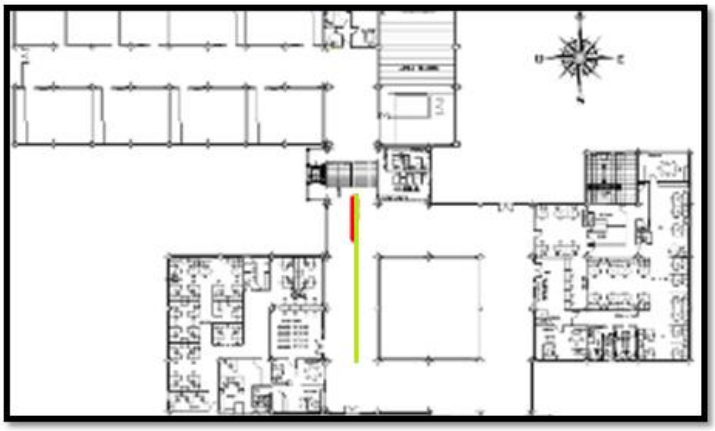

Figura 57. Trayectoria 1 - Detector SIFT

Detector FAST

La Figura 58 muestra la trayectoria estimada con este método. Se obtuvo un error porcentual del 10% en una trayectoria simple rectilínea sin giros. En la Figura 59 se muestra la trayectoria estimada en el Smartphone.

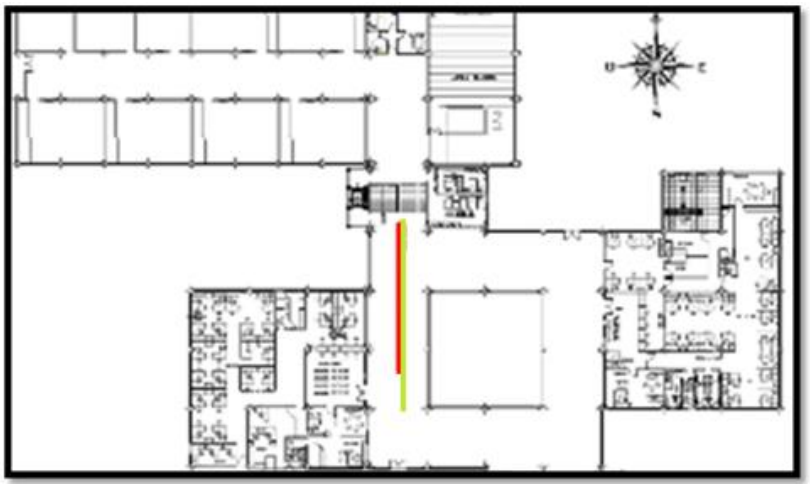

Figura 58 Trayectoria 1 - Detector FAST

Figura 59.Trayectoria 1 - Detector FAST Smartphone.

Detector Harris:

La Figura 60 muestra la trayectoria estimada con el detector de esquinas de Harris. Se obtuvo un error porcentual del 2.5% en una trayectoria simple rectilínea sin giros. En la Figura 61 se muestra la trayectoria estimada en el Smartphone.

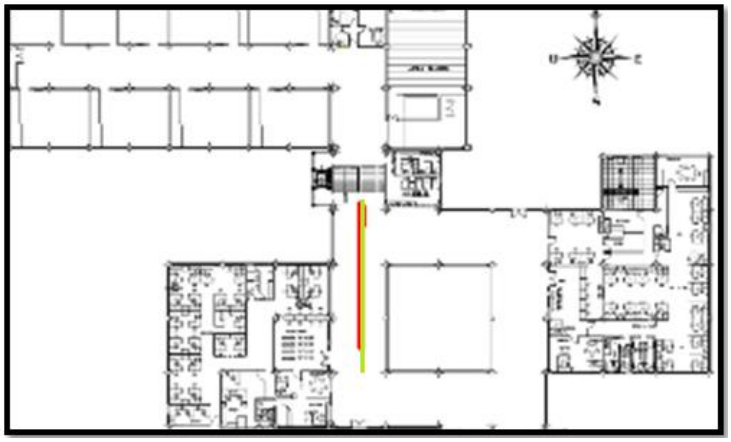

Figura 60 Trayectoria 1 - Detector Harris

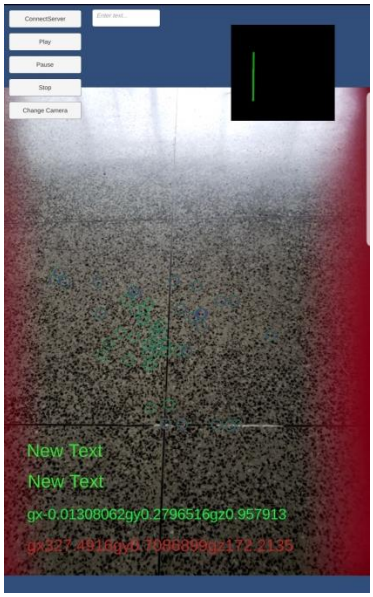

Figura 61. Trayectoria 1 - Detector Harris Smartphone

Detector Shi-Tomasi:

La Figura 62 muestra la trayectoria estimada con el detector de esquinas de Shi-Tomasi. Se obtuvo un error porcentual del 0.2% en una trayectoria simple rectilínea sin giros. En la Figura 63se muestra la trayectoria estimada en el Smartphone.

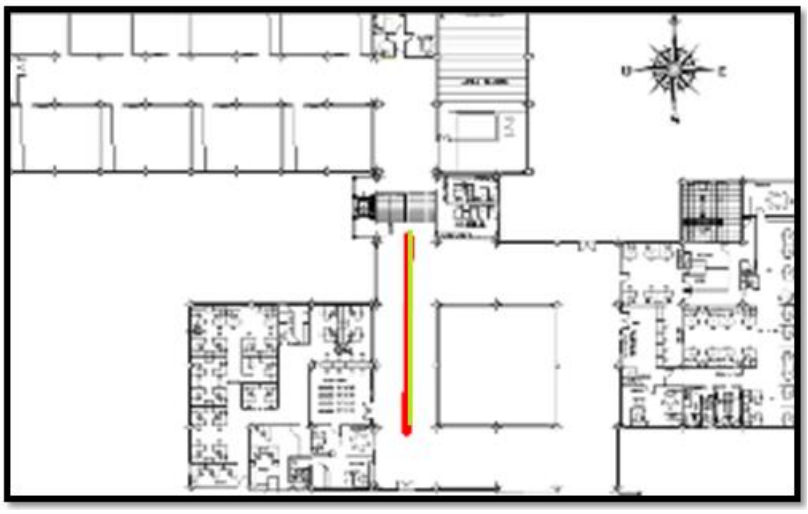

Figura 62 Trayectoria 1 - Detector Shi-Thomas

Figura 63. Trayectoria 1 - Detector Shi-Thomas Smartphone

Error cuadrático medio

La Tabla 8 muestra los valores de la raíz del error cuadrático medio que mide la diferencia entre el estimador y la distancia real.

Tabla 8.
Error cuadrático medio en la trayectoria 1.

	Trayectoria 1			
	SIFT	**FAST**	**HARRIS**	**SHI-THOMASI**
	79	274	238	246
	85	271	244	244
	87	270	242	245
Raíz del Error cuadrático medio	163,3687037	24,7251559	6,191391874	2,160246899

Error relativo y porcentual

En Tabla 9 se muestra el error porcentual de la distancia recorrida versus la distancia estimada en ella se puede observar que el detector SIFT presenta un error porcentual demasiado alto. Mientras que el detector SHI-TOMASI es el más eficiente en la primera prueba realizada.

Tabla 9.
Resultados-Trayectoria 1

Resultados- Trayectoria 1						
Iluminación promedio[lux]	576,66	**Detector**	**Pixeles recorridos**	**Metros estimados[m]**	**Error relativo**	**Error Porcentual (%)**
Pixeles en el plano	247	**SIFT**	85,66	6.69	0,653198381	65,31983806
		FAST	272	21.26	0,101214575	10,12145749
Distancia real[m]	19,31	**Harris**	241	18.84	0,024291498	2,429149798
		Shi-Tomasi	246,5	19.27	0,002024291	0,20242915

- **Segunda trayectoria**

Esta prueba se realizó a las 2 pm con un nivel de iluminación promedio de 166,57 lux. En la segunda prueba se introduce un giro y se aumenta la distancia recorrida.

Detector FAST

La Figura 64 muestra la trayectoria estimada por este detector. Se obtuvo un error porcentual de 7.6%. Se observa claramente como luego del giro la trayectoria estimada se separa de la trayectoria real. La Figura 65 muestra la trayectoria realizada en la pantalla del Smartphone.

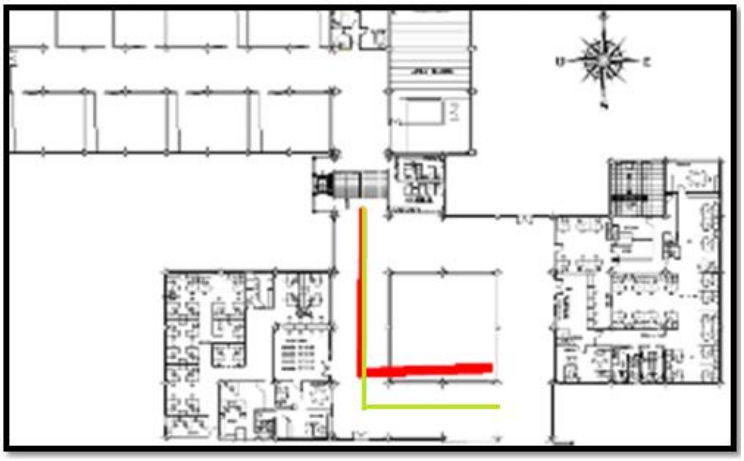

Figura 64 Trayectoria 2 - Detector FAST

Figura 65. Trayectoria 2 - Detector FAST Smartphone

Detector Harris

La trayectoria estimada con el detector de Harris se muestra en la Figura 66. Se observa una diferencia notoria en la parte media de las trayectorias estimada y real, sin embargo finalizan en aproximadamente en el mismo punto. Se obtuvo un error porcentual de 4,17 %. La Figura 67 muestra la trayectoria estimada en el Smartphone.

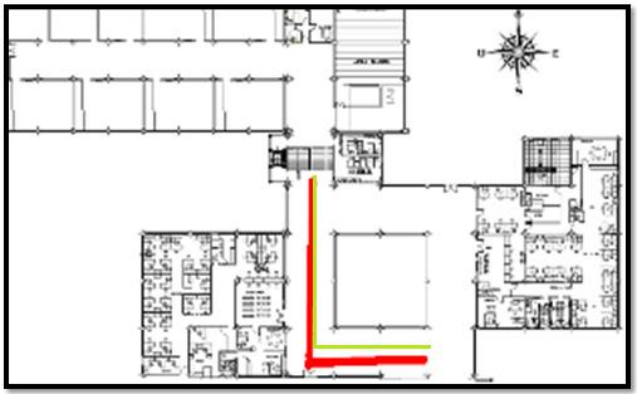

Figura 66 Trayectoria 2- Detector Harris

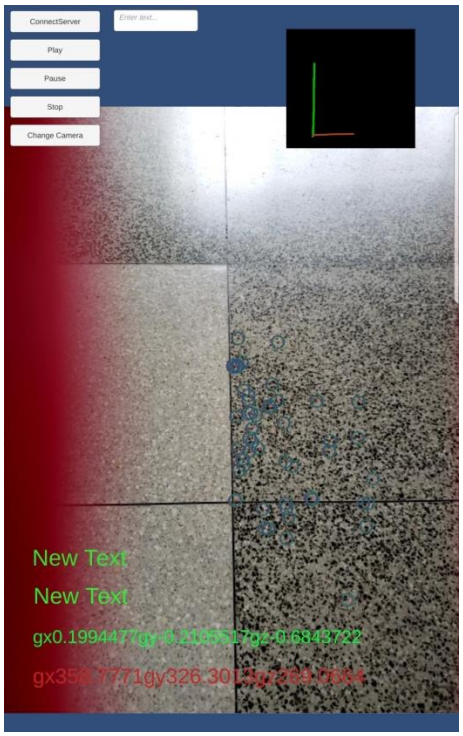

Figura 67.Trayectoria 2- Detector Harris Smartphone

Detector Shi-Tomasi

La trayectoria estimada con el detector de Shi-Tomasi se muestra en la Figura 68.Se observa que a lo largo del desplazamiento sigue la trayectoria real, siembargo al final existe una ligera diferencia de distancias. Se obtuvo un error porcentual de 0,2 %. Figura 69 muestra la trayectoria estimada en el Smartphone.

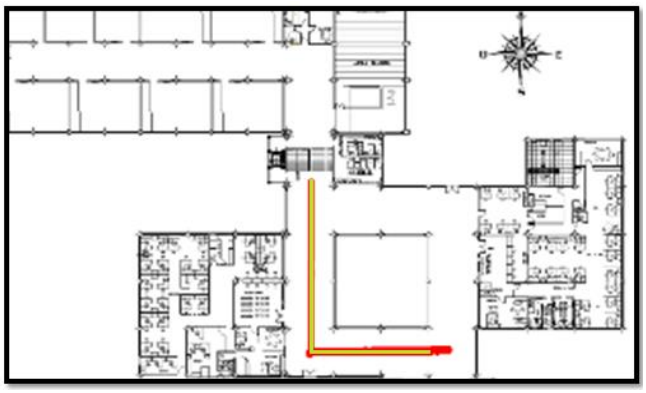

Figura 68 Trayectoria 2 - Detector Shi-Tomasi

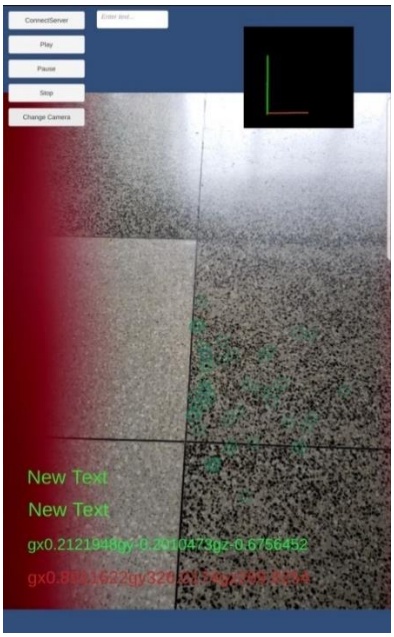

Figura 69. Trayectoria 2 - Detector Shi-Tomasi Smartphone.

Error cuadrático medio

La Tabla 10 muestra los calores del error cuadrático medio presentado en la segunda prueba.

Tabla 10.
Error cuadrático medio en la trayectoria 2.

Trayectoria 2			
	FAST	HARRIS	SHI-THOMASI
	374	390	400
	376	391	406
	375	396	405
Raíz del Error cuadrático medio	32,010415	14,89966443	4,242640687

Error relativo y porcentual

La Tabla 11 muestra los errores relativos y porcentuales, el detector de Shi-Tomasi se presenta como el método más eficiente con un error porcentual de 0.2%

Tabla 11.
Resultados en la trayectoria 2.

Resultados- Trayectoria 2						
		Detector	**Pixeles recorridos**	**Metros estimados[m]**	**Error relativo**	**Error Porcentual (%)**
Iluminación promedio[lux]	16 6,57	**FAST**	376	29.39	0,076167076	7,616707617
Pixeles en el plano	40 7	**Harris**	390	30.49	0,041769042	4,176904177
Distancia real[m]	31, 82	**Shi-Tomasi**	406	31.74	0,002457002	0,245700246

- **Tercera trayectoria**

Esta prueba se realizó a las 4 pm con un nivel de iluminación promedio de 70,57 lux. En la tercera prueba se cambia la trayectoria y se introduce otro giro con el fin de probar la exactitud del sistema.

Detector FAST

La baja iluminación afecta a este método de forma noble, como se observa en la Figura 70 la trayectoria estimada dista mucho de la trayectoria real. Se obtuvo un error porcentual de 35.6%. La Figura 71 muestra la trayectoria estimada en la pantalla del Smartphone.

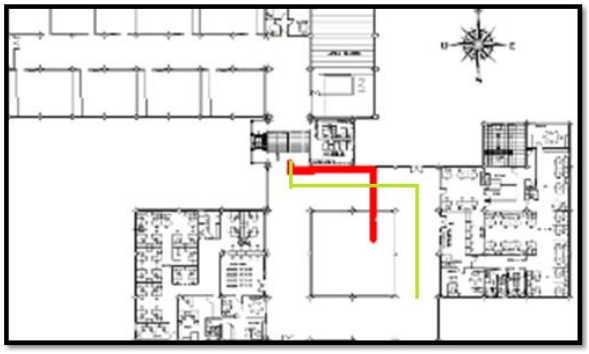

Figura 70 Trayectoria 3 - Detector FAST

Figura 71. Trayectoria 3 - Detector FAST Smartphone

Detector Harris

Apesar de la baja iluminación la Figura 72 muestra apenas una ligera diferencia entre las dos trayectorias. Se obtuvo un error porcentual de 9.7%. La Figura 73 se muestra la trayectoria estimada en la pantalla del smartphone.

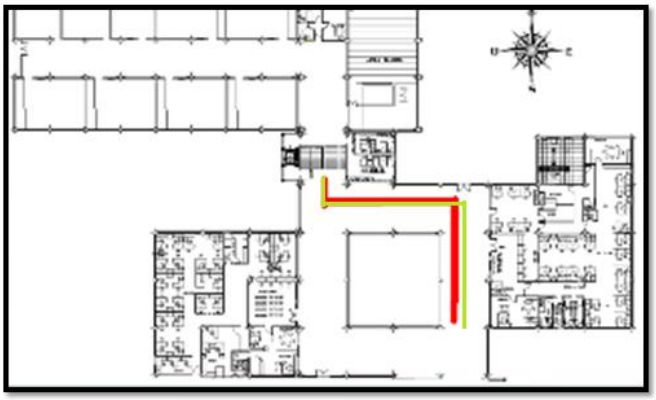

Figura 72 Trayectoria 3 - Detector Harris

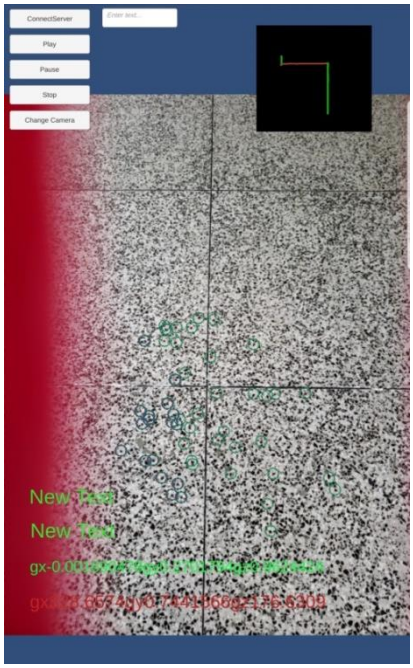

Figura 73. Trayectoria 3- Detector Harris Smartphone

Detector Shi-Thomas

Igual que en las pruebas anteriores la trayectoria estimada por este método resulta ser la más eficiente. Se obtuvo un error porcentual de 1.3 %. La Figura 74 muestra que apenas existe una ligera diferencia entre las dos trayectorias.

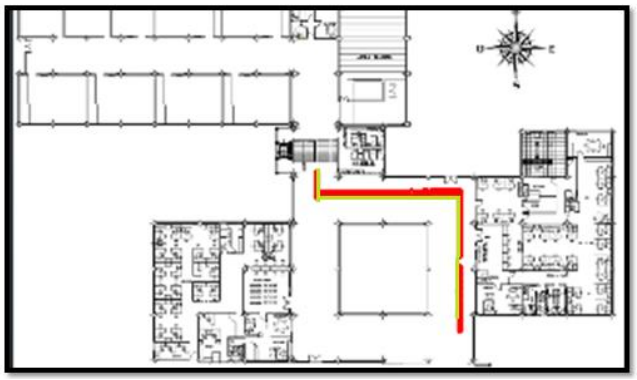

Figura 74 Trayectoria 3 - Detector Shi-Thomas

Figura 75. Trayectoria 3 - Detector Shi-Thomas Smartphone.

Error cuadrático medio

En la Tabla 12 se presenta la raíz del error cuadrático medio obtenido por cada uno de los detectores.

Tabla 12.
Error cuadrático medio en la trayectoria 3.

	Trayectoria 3		
	FAST	HARRIS	SHI-THOMAS
	244	413	389
	240	415	385
	246	411	384
Raíz del Error cuadrático medio	135,689597	34,0391931	7,325753659

Error relativo y porcentual

La Tabla 13 muestra los resultados obtenidos en la tercera prueba. En esta ocasión el detector FAST se presenta como el más deficiente con un 35.6%. De la misma forma que en la prueba anterior SHI-TOMASI es el más eficiente con un error porcentual de 1,3%.

Tabla 13.
Resultados en la trayectoria 3.

Resultados- Trayectoria 3						
		Detector	**Pixeles recorridos**	**Metros estimados[m]**	**Error relativo**	**Error Porcentual (%)**
Iluminación promedio[lux]	70,57	**FAST**	244,04	19.06	0,356094987	35,60949868
Pixeles en el plano	379	**Harris**	415,7	32.466	0,096833773	9,683377309
Distancia real[m]	29,6	**Shi - Tomasi**	384,04	29.99	0,013298153	1,329815303

CAPÍTULO VI

6. CONCLUSIONES Y RECOMENDACIONES

6.1.Conclusiones

La implementación del sistema de estimación de posición y orientación basado en datos de visión y datos inerciales ha sido satisfactoria. Las pruebas realizadas muestran que el sistema implementado puede ejecutarse en tiempo real. Todos los algoritmos de estimación visual-inercial se ejecutan en el dispositivo Smartphone. Los resultados muestran que los detectores de esquinas son más eficientes cuando se trata de aplicaciones en tiempo real, debido a que requieren menos recursos computacionales y su tiempo de ejecución es menor. Hablando puntualmente los detectores de Harris y Shi-Tomasi permiten obtener trayectorias estimadas muy precisas aun cuando los niveles de iluminación del entorno son bajos.

En los experimentos realizados se probaron cuatro detectores de puntos característicos entre ellos se testeó el detector SIFT muy conocido dentro del campo de odometría visual por su robustez y precisión, sin embargo no muy utilizado en aplicaciones en tiempo real debido a que requiere altos niveles de procesamiento. Los resultados de su implementación afirman lo mencionado anteriormente. Con SIFT se obtuvieron errores porcentuales de 65% con diferencias de hasta 15 metros respecto a la distancia real recorrida. El principal problema fue la velocidad de procesamiento (2.8GHz) del Snapdragon 845 presente en el dispositivo móvil. Sin embargo, los algoritmos FAST, Harris y Shi-Tomasi mostraron excelentes resultados a niveles de iluminación alta. Cuando los niveles de iluminación bajan de forma considerable, en los experimentos se registró una variación de 557 lux a 70 lux, el algoritmo FAST presentó deficiencias en la estimación de la trayectoria con una diferencia de 10 metros respecto a la real. Sin embargo los

detectores de Harris y Shi-Tomasi presentaron diferencias de 3 metros para el caso de Harris y apenas 30 cm en el caso del detector de Shi-Tomasi. De esta forma se concluye que el detector ideal para el sistema implementado es el detector de esquinas de Shi-Tomasi debido a su precisión y además permite tener un algoritmo optimizado para un sistema embebido como es un Smartphone.

Los algoritmos de odometría visual permiten estimar la traslación y rotación de la cámara adherida a un agente móvil. Sin embargo, en este sistema se utilizaron únicamente los datos de traslación. Para los giros se utilizaron los datos del giroscopio del dispositivo móvil. Unity al ser una plataforma orientada al desarrollo de aplicaciones móviles, implementa funciones que permite obtener datos compensados de la IMU del dispositivo de tal forma que no es necesaria la calibración del dispositivo. Unity implementa filtros pasa bajos para obtener datos libres de ruido.

El punto débil de este sistema es el método de comunicación ente el Smartphone y el computador donde se realiza la simulación de la trayectoria en un entorno 3D. La red implementada es de tipo cliente-servidor, a través de una conexión Wi-fi. Como se conoce una red de comunicación directa entre un ordenador y otro dispositivo sin elementos como routers no permiten alcanzar grandes distancias. Es por esta razón que en este proyecto de investigación se plantea el uso de una red GPRS como solución. De modo que se pueda utilizar la red celular y de esta forma alcanzar distancias mayores.

6.2.Recomendaciones

Se realizaron pruebas del sistema sobre distintos dispositivos móviles. Para obtener datos exactos se recomienda el uso de un dispositivo con un procesador con velocidad de reloj mayor a 2GHz, que se pueden encontrar en Smartphons de gama media alta.

Un método para optimizar el rendimiento del sistema de estimación es el reducir el tamaño de la imagen capturada. Por defecto el sistema captura imágenes de 640x480 píxeles a 30 fps. De modo que al reducir el tamaño de la imagen a 320x240 píxeles se libera espacio en la GPU del procesador y se mejora el rendimiento de la aplicación.

La aplicación desarrollada es compatible con todos dispositivos Android con sistema operativo Android 4.0 y superiores.

Al momento de ejecutar el sistema de estimación se debe mantener el teléfono con una inclinación fija de 30 grados respecto al eje Y. De modo que se conserve la escala entre las imágenes capturadas por la cámara del dispositivo y que los resultados sean confiables.

REFERENCIAS BIBLIOGRÁFICAS

Pretexsa.(Mayo 2018).*¿Por qué no funciona el GPS dentro de un edificio?*Recuperado de http://www.pretexsa.com/RMRv9D2M.html

Kamlofsky,A,Bergamini M. (2014). *Los cuaterniones en visión robotica.*Universidad Abierta Interamericana.

OpencCV team.(2016).*OpenCV library*. California. Recuperado de https://opencv.org/about.html?fbclid=IwAR26MSSVYe8Ty6eby_9TOx3D-jZrP7RSWUEYmjHpsTYhvD9ccm3_LCkdXzM

Ágila, J., Link, B., Smith, P., & Wehrle, K. (2013). *Indoor Navigation on Wheels (and Without) using Smartphones*.Sydney.IEEE Xplore

Aguilar, W., Casaliglla, V., & Pólit, J. (2017). *Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles*. *Electronics*.

Aguilar, W. G., Abad, V., & Ruiz, H. (2018). *RRT-Based Path Planning for Virtual Bronchoscopy Simulator*. En Lecture Notes in Computer Science (págs. 155-165).

Aguilar, W. G., & Angulo Bahón, C. (2013). *Estabilización robusta de vídeo basada en diferencia de nivel de gris.* Memorias Del VIII Congreso de Ciencia y Tecnología. Sangolquí, Ecuador.

Aguilar, W. G., & Angulo, C. (2012). *Compensación y Aprendizaje de Efectos Generados en la Imagen durante el Desplazamiento de un Robot.* X Simposio CEA de Ingeniería de Control. Barcelona, Spain.

Aguilar, W. G., & Angulo, C. (2013). *Autonomous Navigation Control for Quadrotors in Trajectories Tracking*. En Lecture Notes in Computer Science (págs. 287-297).

Aguilar, W. G., & Angulo, C. (2014a). *Estabilización de vídeo en micro vehículos aéreos y su aplicación en la detección de caras.* IX Congreso de Ciencia y Tecnología ESPE. Sangolquí, Ecuador.

Aguilar, W. G., & Angulo, C. (2014b). *Real-time video stabilization without phantom movements for micro aerial vehicles.* EURASIP Journal on Image and Video Processing , 1, 1-13.

Aguilar, W. G., & Angulo, C. (2014c). *Robust Video Stabilization based on Motion Intention for Low-Cost Micro Aerial Vehicles.* 11th International Multi-Conference on Systems, Signals & Devices (SSD). Barcelona, Spain.

Aguilar, W. G., & Angulo, C. (2015). *Real-Time Model-Based Video Stabilization for Microaerial Vehicles*. Neural Processing Letters , 43 (2), 459-477.

Aguilar, W. G., Casaliglla, V. P., & Polit, J. L. (2017). *Obstacle Avoidance for Low-Cost UAVs.* IEEE 11th International Conference on Semantic Computing (ICSC). San Diego.

Aguilar, W. G., Costa-castelló, R., & Angulo, C. (2014). *Control Autónomo de Cuadricopteros*

para Seguimiento de Trayectorias. IX Congreso de Ciencia y Tecnología ESPE. Sangolquí, Ecuador.

Aguilar, W. G., Luna, M. A., & Moya, J. F. (2018). *Real-Time Detection and Simulation of Abnormal Crowd Behavior*. En Lecture Notes in Computer Science (págs. 420-428).

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Parra, H., & Ruiz, H. (2017). *Pedestrian Detection for UAVs Using Cascade Classifiers with Meanshift.* IEEE 11th International Conference on Semantic Computing (ICSC). San Diego.

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Lopez, W. (2018). *Cascade Classifiers and Saliency Maps Based People Detection.* En Lecture Notes in Computer Science (págs. 501-510).

Aguilar, W. G., Luna, M. A., Ruiz, H., Moya, J. F., Luna, M. P., Abad, V., & Parra, H. (2013). *Statistical Abnormal Crowd Behavior Detection and Simulation for Real-Time Applications*. En Lecture Notes in Computer Science (págs. 671-682).

Aguilar, W. G., Morales, S., & Ruiz, H. (2018). *RRT* GL Based Path Planning for Virtual Aerial Navigation*. En Lecture Notes in Computer Science (págs. 176-184).

Aguilar, W. G., & Rodr, G. A. (2018). *Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing.* En Lecture Notes in Computer Science (págs. 410-419).

Aguilar, W. G., Rodríguez, G. A., & Álvarez, L. (2013). *On-Board Visual SLAM on a UGV Using a RGB-D Camera*. En Lecture Notes in Computer Science (págs. 298-308).

Aguilar, W., & Morales, S. (2016). *3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms.* Electronics , 5 (4), 70.

Amaguaña, F., Collaguazo, B., & Tituaña, J. (2018). *Augmented Reality, Virtual Reality, and Computer Graphics.* International Conference on Augmented Reality, Virtual Reality and Computer Graphics (págs. 394-403). Springer.

Armesto, L., Tornero, J., & Vincze, M. (2007). *Fast Ego-motion Estimation with Multi-rate Fusion of Inertial and Vision 1*.The International Journal of Robotics.Sage Journal

Asociados, M., World Meteorological Organization, & Wmo. (2005). *Guide to Meteorological Instruments and Methods of Observation: Part III - quality assurance and management of observing systems.*American Mathematics Society

Bevilacqua, M., Tsourdos, A., & Starr, A. (2016). *Egomotion estimation for monocular camera visual odometer.*IEEE International Instrumentaction and Measurement Technology Conference.

Serrano,E,Sirme,R,La Mura,G. (2015).*Rotaciones y Cuaternios*.Ciencia y Tecnologia. Escela Superior Técnica-IUE.

Derpanis, K. G. (2010). *Overview of the RANSAC Algorithm*.Coputer Science. York University

Rodrigues,J. (2015). *Un Sistema de Odometría Visual Monocu lar bajo Restricciones de Tiempo*

Real.Departamento de Informática yAutomática. Universidad de Salamanca.

Herrera,D.(2014).*Edificio Central | 3D Warehouse*. Recuperado de https://3dwarehouse.sketchup.com/model/45857c07d7a01db8a0ae020ba5c5567c/Edificio-Central

En, R., Atica, A., & De, I. (2015). *Los cuaterniones y su importancia en la representación gráfica por ordenador*. Matemática Computacional.Universitat Jaume.

OpenCV Team. (2010). *FAST Algorithm for Corner Detection*. California. Feature Detection and Description. Recuperado de: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html

Blender Developers Blog. (2013).*FBX binary file format specification*.Recuperado de https://code.blender.org/2013/08/fbx-binary-file-format-specification/

Flores, P., & Braun, J. (2011). *Algoritmo SIFT : fundamento teórico*.Instituo de Eléctrica.Universidad de la Republica.

Fraundorfer, B. F., & Scaramuzza, D. (2012). *Visual Odometry*.IEEE Robotics and Automation Magazine.

Sinha,U (2010).*Fundamentals of Features and Corners: Harris Corner Detector*.AI Shack. Recuperado de http://aishack.in/tutorials/harris-corner-detector/?fbclid=IwAR0jOxjroL3jTmyialdaT_YA1paCI7MFGyKAUp0kkcJ6uRc8Ik1ZzCJ9f14

Garzon,J(2018).*Galaxy S9 Plus: características*.Cnet Recuperado de https://www.cnet.com/es/analisis/samsung-galaxy-s9-plus-opiniones/

Guallichico, U. A., & Ávalos, J. R. A. (2013). *Diseño e Implementación de un Sistema de Navegación Inercial Tipo Strapdown para estimar la Posición de un Robot Móvil* .Revista Politécnica.Quito-Ecuador.

Hannuksela, J., Barnard, M., Sangi, P., & Heikkilä, J. (2011). *Camera-Based Motion Recognition for Mobile Interaction.* ISRN Signal Processing.

OpenCV team (2014).*Harris Corner Detection — OpenCV 3.0.0.* California.Recuperado de https://docs.opencv.org/3.0beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html?fbclid=IwAR26MSSVYe8Ty6eby_9TOx3DjZrP7RSWUEYmjHpsTYhvD9ccm3_LCkdXzM

Hofmann-Wellenhof, B., Lichtenegger, H., & Collins, J. (2001). *Global Positioning System : theory and practice*. Springer-Verlag.

Huang, T. S., & Netravali, A. N. (1994). *Motion and Structure from Feature Correspondences : A Review I* .Proceedings of the IEEE

Cruz,J,Martínez, R.,Pérez, X. (2014). *Implementación del detector y descriptor SURF en dispositivos móviles con sistema operativo Android*. Congreso Internacional de Investigación.Academia Journal .México

Lacey, A. J., Pinitkarn, N., Thacker, N. A., Lacey, A. J., Pinitkarn, N., & Thacker, N. A. (2002). *An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration.*BMVC2000.Manchester.Oxford

Lytton, W. W. (2014). *Developing of a Video-Based Model for UAV Autonomous Navigation.* Encyclopedia of the Neurological Sciences, *4*, 844–847.

Monreal,F.(2015).*Manual de Operaciones de Combate en Áreas Urbanas*. Ministerio de defensa .Venezuela.Recuperado de https://issuu.com/uvesociety/docs/manual_de_operaciones_de_combate_en?fbclid=IwAR22qC5oDTHIjoPiRbs3FAEjE-hwkvJNvyGuCEZ4HNf8jGdgFZ_uqUkyYVM

Markelov, P. A. (2002). *Uso de Cuaterniones para Representar Rotaciones.*Revistas Académicas UTP.Tecnología Hoy.

Mentor, A., & Programaci, S. (2012). *Desarrollo de aplicaciones para Android I*.Aula Mentor.

Nist, D. (2004). *An Efficient Solution to the Five-Point Relative Pose Problem 3 . The Five-Point Algorithm*.IEEE Transactions on Pattern Analysis and Machine Intelligence

Nist, D., & Bergen, J. (2011). *Visual Odometry*.IEEE Robotics and Automation Magazine

OpenCV team(2018).*OpenCV: Camera Calibration.* California. Recuperado de https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html?fbclid=IwAR3Na8aF2ZgIaP2RGYOimWcgc0CjHLEAXlfR32dYn9SKtimpkat7z_Kvp7c

Opsahl, T. (2016). *The perspective camera model*.Universidad de Oslo

Patel, D. (2013). *Optical Flow Measurement using Lucas kanade Method*.International Journal of Computer Application.

Rister, B., Wang, G., Wu, M., & Cavallaro, J. R. (2013). *A FAST and Efficient SIFT Detector using the mobile GPU*. Cavallaro Department of Electrical and Computer Engineering , Rice University , Houston , Texas. Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference On.

Rojas, I., Moncusi, J. C. I., & Joya, G. (2013a). *Obstacle Avoidance for Flight Safety on Unmanned Aerial Vehicles*. Soft Computing.

Rojas, I., Moncusi, J. C. I., & Joya, G. (2013b). *Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps.* Soft Computing.

Rojas, I., Moncusi, J. C. I., & Joya, G. (2013c). *RRT* GL Based Optimal Path Planning for Real-Time Navigation of UAVs.* Soft Computing.

Rojas, I., Moncusi, J. C. I., & Joya, G. (2013d). *Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing*. Soft Computing.

Scaramuzza, B. D., & Fraundorfer, F. (2011). *Visual Odometry*, (December).IEEE Robotics and Automation Magazine.

Scaramuzza, D., Fraundorfer, F., & Siegwart, R. (2009). *Real-Time Monocular Visual Odometry for On-Road Vehicles with 1-Point RANSAC*. IEEE International Conference on Robotics and Automation.Kobe-Japón

Samsung(2018).*Samsung Galaxy S9 and S9+*. Recuparado de https://www.samsung.com/global/galaxy/galaxy-s9/specs/

Szeliski, R. (2010). *Computer Vision : Algorithms and Applications.*Springer.

Tomasi, J. S. y C. (1994). *Good features to track.*IEEE Coference and Computer Vision and Pattern Reconigtion. Seattle.

Under, N. O. T., & Control, R. (2011). *Making Android Motion Applications Using the Unity 3D Engine*.InvenSense.

Unity docs (2016). *Manual: Creando y usando scripts.* California. Scripting API.Recuperado de https://docs.unity3d.com/es/current/Manual/CreatingAndUsingScripts.html

Unity docs(2016). *Manual: Execution Order of Event Functions (Orden de Ejecución de Funciones de Evento).* California. Scripting API.Recuperado de https://docs.unity3d.com/es/current/Manual/ExecutionOrder.html

Unity docs (2016). *Manual: Formatos 3D.* California. Scripting API. Recuperado de https://docs.unity3d.com/es/current/Manual/3D-formats.html

Unity docs (2018). *Manual: Input.* California. Scripting API. Recuperado de https://docs.unity3d.com/Manual/Input.html

Unity docs (2018). *Manual: Physics.* California. Scripting API. Recuperado de https://docs.unity3d.com/Manual/PhysicsSection.html

Unity docs (2018). *MonoBehaviour.Awake().*California. Scripting API. Recuperado de https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html?_ga=2.114835265.753653915.1537855101-1416430817.1510872674

Unity docs (2018). *MonoBehaviour.Start().*California. Scripting API. Recuperado de https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html?_ga=2.10342515.753653915.1537855101-1416430817.1510872674

Unity docs (2018). *Quaternion.* California. Scripting API. Recuperado de https://docs.unity3d.com/ScriptReference/Quaternion.html

Unity docs (2016). *Rigidbody.* California. Scripting API. Recuperado de https://docs.unity3d.com/es/530/ScriptReference/Rigidbody.html?fbclid=IwAR13GBQQDDRIiz86Jfm4zV0Kh2K6GR-gCemXYu6LVyv17guFCnBthtok7X8

Vedaldi, A. (2007). *An open implementation of the SIFT detector and descriptor*. UCLA CSD Technical Report *070012*, *1*, 1–6.

Villca, C. (2017). *Algoritmo SIFT para la detección de imágenes coincidentes.*Revista de Investigacion Estudiantil Iluminate

Woodman, O. J. (2007). *An introduction to inertial navigation*, (696).Computer Laboratory Technical Reports.Univertisy of Cambrige.

Yamaguchi, K. (2006). *Vehicle Ego-Motion Estimation and Moving Object Detection using a Monocular Camera, 18–21.*International Conference on Pattern Recognition

Yamamoto, S., Mae, Y., Shirai, Y., & Miura, J. (1995). *Realtime Multiple Object Tracking Based on Optical Flows.*IEEE International Conference Robotics on Automation